COMPUTER PROGRAMMING

Using

VB.NET

for the subject

INTRODUCTION TO

COMPUTER PROGRAMMING

FET NCV NQF Level 3

(TOPICS 2,3,4,6)

Notes prepared by:

HG ERASMUS

CM PRETORIUS

© COPYRIGHT: THE AUTHORS

All rights reserved.

These notes remain the intellectual property of the authors.
Copyright assigned to FET-colleges and to the DoE to duplicate the document in hard copy format only. Any other reproduction method is prohibited. No changes or alterations, in any way, are permitted.

The authors of each section of these notes take full responsibility regarding copyright issues pertaining to the individual content developed.

Suggestions and comments may be e-mailed to:

Hetsie@telkomsa.net

PREFACE

These level 3 notes are a follow up to the Level 2 notes and were also developed to cover the missing outcomes from the amended NCV-statement for the subject INTRODUCTION TO SYSTEMS DEVELOPMENT.

The notes reflect the interpretation of the authors regarding the NCV-document. It remains the responsibility of the lecturer to ensure that all subject outcomes in the NCV-document are covered.

The primary focus in the notes is on Visual Basic.NET application development and the following major subject outcomes are covered:

- String manipulation
- Function produces and independent sub procedures
- One and two dimensional arrays
- Sequential text files
- Advanced features and functions such as random, dates and mathematical functions
- Comprehensive examples combining all the topics
- Error handling and debugging techniques.

The notes also contain 2 appendixes and a complete set of exercises and sample solutions are also included to assist lecturers. Appendix A provides a summary on user interface and output design concepts that have been covered in Level 2 notes, but applied in level 3 notes. Appendix B provides an introduction to databases.

Various examples have been supplied. The solution of these problems was planned via a method to identify input and output variables and algorithms. The VB.NET code and examples of output are also provided. Due to the fact that every example illustrates a new concept, these programs do not necessarily include data validation and input handling. These concepts were included in the comprehensive examples in chapter 6.

In VB.NET, the procedure header for every button click event is a very lengthy statement such as

```
Private Sub btnClear_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnClear.Click
```

Therefore, in every program example, the authors have made use of the ellipse as a method to save space and to fit the procedure header onto one line.

```
Private Sub btnClear_Click(. . .) Handles btnClear.Click
```

The following icons, used in level 2 notes, are also used in these notes:



Questions

This icon will be followed by short questions that are suitable for class discussions. Sometimes these questions are used to refresh your memory on previous chapters and other times questions about the newly discussed content.



Exercises

This icon will be followed by exercises that the student must solve on paper. This could possibly lead to class discussions at a later stage, but students must first think about the problems and solve it on paper by applying the knowledge they have learned.



Practical

This icon precedes practical assignments that must be solved in Visual Basic.NET. Some of these assignments may already have been planned in previous exercises, but others must be planned and designed from scratch. All of these solutions must be tested with test data.

We hope you enjoy these topics and that it will inspire you to use advanced concepts and methods in VB.NET to solve any problem which you may encounter!

TABLE OF CONTENTS

CHAPTER		PAGE
1	String manipulation	1
2	Function Procedures And Sub Procedures	23
3	Arrays	50
4	Sequential text files	86
5	Advanced features and build-in functions	115
6	Comprehensive examples	132
7	Error handling and debugging techniques	153
Questions and Answers		168

APPENDIXES

A	GUI design principles	209
B	Introduction to DB concepts and the ADO.NET DB wizards	293

CHAPTER 1
STRING MANIPULATION

1.1 INTRODUCTION

A string consists of two or more characters that must be enclosed in quotes.

Examples: "32 Long Street" and a message such as "The name is found".

A variable my also be of the data type *string* as seen in the following example:

```
Dim  firstName As String
```

where firstName may contain a string value.

There are many reasons why strings are manipulated e.g. a unique code is created by using different fields, a description may be changed according to prescribed rules, a part of a string may have a specific meaning and must be tested, etc.

In Visual Basic.NET there are various ways to manipulate strings, and the ones most commonly used, will be discussed.

OUTCOMES

When you have studied this chapter you should be able to know, understand and use the following ways to manipulate strings in VB.NET:

- Join two or more strings by means of the Concatenation operator
- Length property
- IndexOf method
- Replace method
- Mid statement
- Remove method
- Trim method
- TrimStart method
- TrimEnd method
- Insert method
- PadLeft method
- PadRight method
- Substring method
- Compare method
- StartsWith method
- EndsWith method
- Contains method

1.2 JOIN TWO OR MORE STRINGS INTO ONE STRING

Joining or linking strings together is called **concatenation** and the symbol used to concatenate 2 or more strings is the ampersand **&** character.

Example 1.2.1

If **intResult** is an integer variable that contains the number 17, the following statement will produce the given output on the label:

Statement:

```
lblAnswer.Text = "The answer is " &  intResult.ToString
```

Output on the label: The answer is 17

Example 1.2.2

If the variables **strName** and **strLastName** of the data type string contain the words Jocelyn and Jones respectively, the following statement will produce the given output on the label:

Statement:

```
lblAnswer.Text  = "The name of the girl is " & strName & _
                  " whose surname is "  & strLastName & "."
```

Output: The name of the girl is Jocelyn whose surname is Jones.

1.3 DETERMINE THE NUMBER OF CHARACTERS IN A STRING

The **length property** can be used to determine the number of characters contained in a string.

General format: *string*.Length

Example 1.3.1

The string "Example" contains 7 characters, therefore

```
"Example".Length = 7
```

Example 1.3.2

The space is also counted as a character, therefore

```
"VB.NET course".Length = 13
```

Example 1.3.3

Write the code to calculate the sum of the length of the string variable **strDescription** and the value of the integer variable called **intNum12**.

```
intSum = strDescription.Length + intNum12
```

Example 1.3.4

The employees of a company all have 8 character personnel numbers. You are asked to write part of an algorithm for the following: Enter this number and then test if it contains exactly 8 characters. If not, display a suitable message.

Algorithm:

```
display "Enter the personnel number"
enter persNumber
if persNumber.Length = 8
        display persNumber, " is correct"
else
        if persNumber.Length < 8
                display persNumber, " contains less than 8 characters"
        else
                display persNumber, " contains too many characters"
        endif
endif
```

The messages in the previous code can also indicate how many characters the personnel number contains e.g. if it contains less numbers the display statement could be:

```
        display persNumber, " contains less than 8 characters = ", persNumber.Length
```

1.4 INDEX OF A CHARACTER IN A STRING

The **index** of a character in a string indicates the position of the character in the string. The first character has an index of 0 while the last character has an index of (n - 1) where n = number of characters in the string.

It is possible to determine the index of a specific character by using the **IndexOf method**. This method can also determine whether a string contains a specific sequence of characters (indicated by the argument *substring*) in which case it will return the starting position of these characters. The argument *beginIndex* is optional. If omitted, the search will begin at the first character in the string otherwise, the search will begin at this position:

General format: s*tring*.IndexOf(substring[, beginIndex])

Note that the value of an index is always an integer.

Example 1.4.1:

```
Dim strValue As String = "AB12345"
Dim intIndex As Integer

intIndex =   strValue.IndexOf("B")
```

intIndex will now contain the value of 1
(A is at index position 0 and B is found at index position 1)

Example 1.4.2

```
Dim strValue As String = "ABABAB12345"
Dim intIndex As Integer

intIndex = strValue.IndexOf("B", 4)
```

intIndex will now contain the value of 5 because the first occurrence of a B after index position 4, is position 5.

Example 1.4.3:

```
Dim strValue As String = "AB12345"
Dim intIndex As Integer

intIndex = strValue.IndexOf("C")
```

intIndex will now contain the value of –1 because the value C was not found.

A value of –1 will also indicate the end of a string.

1.5 REPLACE CHARACTERS IN A STRING

1.5.1 THE REPLACE METHOD

The **replace method** can replace all occurrences of one or more characters in a string by one or more other characters.

General format: *string*.Replace (firstValue, secondValue)

This method will replace the firstValue in *string* by the secondValue. Note that the first- and the secondValue need not have the same number of characters.

Example 1.5.1.1:

```
Dim strDescription As String = "abcdefg"
Dim strResult As String

strResult = strDescription.Replace("cdef", "xx")
```

The value of strResult will now be "abxxg"

1.5.2 THE MID STATEMENT

The **Mid statement** can also replace a specified number of characters in one string with characters from another string.

General format: Mid (firstString, start [, count]) = secondString

firstString is the string in which the characters must be replaced. The string that contains the replacing characters is *secondString*. The character position of the first character to be replaced in firstString is *start*. The first character in the second string is character position 1, the next character is at character position 2 etc. Take note that this is not the same as the index. The first position of the index is always 0.
The optional argument, *count*, specifies the number of characters to be replaced in *firstString*. If it is omitted, the lesser of either the number of characters in *firstString* or *secondString* will be replaced from position *start* to the end of *firstString*.

Example 1.5.2.1:

```
Dim strName As String = "Mpho Kgoebane"
Dim strNewChar As String = "e"

Mid (strName, 11, 1) = StrNewChar
```

The value of strName will now be "Mpho Kgoebene"

Example 1.5.2.2:

Because the value of strNewChar is only one character, the following statement will result in the same answer.

```
Dim strName As String = "Mpho Kgoebane"
Dim strNewChar As String = "e"

Mid (strName, 11) = StrNewChar
```

The value of strName will now also be "Mpho Kgoebene"

Example 1.5.2.3:

```
Dim strName As String = "Mpho Kgoebene"
Dim strNewChar As String = "lani"

Mid (strName, 12) = StrNewChar
```

The value of strName will now be "Mpho Kgoebela" as the count option has not been specified and the lesser of number of characters have been taken as the count. In the first string there were 2 characters from position 12 onwards ("ne") and in the second string there were 4 characters ("lani"). Only two characters were therefore replaced.

Example 1.5.2.4:

```
Dim strName As String = "Mpho Kgoebene"
Dim strNewChar As String = "lani"

Mid (strName, 12, 4) = StrNewChar
```

Although it seems as if 4 characters must be replaced and the value of strName must now be "Mpho Kgoebelani", these statements will also result in "Mpho Kgoebela" as there are only two characters from position 12 that can be replaced.



Questions

Can any Mid-statement be rewritten by applying the replace method to the string? Will it produce the same result?

1.6 REMOVE CHARACTERS FROM A STRING

1.6.1 THE REMOVE METHOD

The **Remove method** allows the user to remove one or more characters from anywhere in a string.

General format: *String*.Remove (beginIndex, count)

The position of the first character in the string to be removed is indicated by *beginIndex* and *count* represents the number of characters to be removed.

Example 1.6.1.1:

```
Dim strCode As String = "abcd76543"
Dim strAnswer As String

strAnswer = strCode.Remove(3,2)
```

The value of strAnswer will now be "abc6543" as the d in the value is at index position 3 and 2 characters are removed.

Example 1.6.1.2:

```
Dim strCode As String = "abcd76543"
Dim strResult As String

strResult = strCode.Remove(7,2)
```

The value of result will now be "abcd765" as the 4 in the value is at index position 7 and 2 characters are removed.

1.6.2 THE TRIM METHOD

The **Trim method** can be used to remove the specified character or characters from both the beginning and end of a string. If no character is specified, the method will take a space as the default character and therefore all leading spaces and trailing spaces will be removed.

General format: *String*.Trim(characters)

Example 1.6.2.1:

```
Dim strWord1 As String = "  13.5%    "
Dim strWord2 As String
strWord2 = strWord1.Trim()
lstDisplay.Items.Add(strWord1)
lstDisplay.Items.Add("Length = " & strWord1.Length)
lstDisplay.Items.Add(strWord2)
lstDisplay.Items.Add("Length = " & strWord2.Length)
```

The following will be displayed in the list box:
  13.5%
Length = 11
13.5%
Length = 5

No character is specified and therefore all leading and trailing spaces will be removed from strWord1. It contains 2 leading and 4 trailing spaces which will be removed by the trim method. Therefore the length of the strword1 is 11 and the length of the strWord2 is 5 characters.

Example 1.6.2.2:

```
Dim strWord1 As String = "  13.5%    "
Dim strWord2 As String
strWord2 = strWord1.Trim("%"C, " "C)
lstDisplay.Items.Add(strWord1)
lstDisplay.Items.Add("Length = " & strWord1.Length)
lstDisplay.Items.Add(strWord2)
lstDisplay.Items.Add("Length = " & strWord2.Length)
```

In this example, it is specified that all leading and trailing spaces must be removed, but that any leading or trailing percentage signs must then also be removed. Therefore, the following will be displayed in the list box:

```
  13.5%
Length = 11
13.5
Length = 4
```

TAKE NOTE: "%"C instead of "%" is used, because it is a character, not a string.

```
Dim strWord1 As String = "  13.5%    "
Dim strWord2 As String
strWord2 = strWord1.Trim("%")
lstDisplay.Items.Add(strWord1)
lstDisplay.Items.Add("Length = " & strWord1.Length)
lstDisplay.Items.Add(strWord2)
lstDisplay.Items.Add("Length = " & strWord2.Length)
```

In this example, it is specified that only leading or trailing percentage signs must be removed. Because the percentage sign is not a trailing character (it is followed by 4 spaces), nothing will be removed from the string and the following will be displayed in the list box:

```
  13.5%
Length = 11
  13.5%
Length = 11
```

1.6.3 THE TRIMSTART METHOD

The **TrimStart method** can remove characters from the beginning of a string. Once again, if no character is specified, it will remove all leading spaces in the string.

General format: *String*.TrimStart(characters)

Example 1.6.3.1:

```
Dim strWord1 As String = "       beauty"
Dim strWord2 As String

strWord2 = strWord1.TrimStart(""C)
```

In this example, all the leading spaces will be removed and the value in strWord2 will be “beauty”.

Example 1.6.3.2:

```
Dim strWord1 As String = "       beauty"
Dim strWord2 As String

strWord2 = strWord1.TrimStart()
```

This example will achieve the same result as the one in example 1.6.3.1.

1.6.4 THE TRIMEND METHOD

The **TrimEnd method** can remove characters from the end of a string and if no character is specified, all spaces at the end of the string will be removed.

General format: *String*.TrimEnd(characters)

Example 1.6.4.1:

In the following example, the % is removed from the string and the remaining value is converted to a decimal value before it is placed in a decimal variable called decAmount:

```
Dim strAmount As String = "12.5%"
Dim decAmount As Decimal

decAmount = Convert.toDecimal(decAmount.TrimEnd("%"C))
```

Example 1.6.4.2:

In the following example, only the trailing asterisks will be removed from the string and the value in strNewValue will be **52.

```
Dim strOldValue As String = "**52**"
Dim strNewValue As String

strNewValue = strOldValue.TrimEnd("*"C)
```

1.7 INSERT CHARACTERS WITHIN A STRING

1.7.1 INSERT METHOD

The **Insert method** is used to insert one or more characters anywhere in a string. The argument *beginIndex* specifies the value of the index where the insertion process must start, while *value* specifies the string value that must be inserted at that position.

General format: *String*.Insert(beginIndex, value)

Example 1.7.1.1:

```
Dim strCode As String = "abcd76543"
Dim strAnswer, strResult As String

strAnswer = strCode.Insert(0, "rst")
strResult = strAnswer.Insert(2, "23")
```

The value of strAnswer will be "rstabcd76543" as the value "rst" is inserted from the first position of the value (Index position 0).

Note that the value of strResult will now insert a value into the string variable strAnswer. The value of strResult = "rs23tabcd76543".

1.7.2 PADLEFT METHOD

The **PadLeft method** can be used to insert characters in the beginning of the string. By inserting the padding characters at the beginning of the string, implies that the characters in the string will be right aligned.

General format: *String*.PadLeft(length[, character])

The desired length of the string can be specified by *length*, while *character* indicates the padding character. If the character is omitted, a space will be the default character.

The PadLeft method is normally used to align numeric values when more than one value must be displayed underneath one another. Remember, that a numeric value will first have to be converted to a string value before the PadLeft method can be applied to it.

Example 1.7.2.1:

```
Dim strWord As String = "Hello"
strWord = strWord.PadLeft(20, "*"C)
```

The new value of strWord will now be `**********Hello`

Example 1.7.2.2:

```
Dim decNumber1 As Decimal = 12.99D
Dim decNumber2 As Decimal = 75556.85D
lstDisplay.Items.Add = decNumber1.ToString("N2").PadLeft(10)
lstDisplay.Items.Add = decNumber2.ToString("N2").PadLeft(10)
```

The following will be displayed in the list box:

```
     12.99
 75,556.85
```

1.7.3 PADRIGHT METHOD

The **PadRight method** can be used to insert characters at the end of the string. By inserting the padding characters at the end of the string, implies that the characters in the string will be left aligned.

General format: *String*.PadLeft(length[, character])

If the first names of employees and the surnames of employees must be displayed in two columns, the PadRight method can be used to display the names as from position 1 and to fill every first name up with spaces until the length of the first name is 20 characters. The surname can then be displayed which will imply that all surnames will start at position 20 as indicated in the next example:

Example 1.7.3.1:

```
Dim strFirstName As String = "Sally"
Dim strSurname As String = "Jones"
lstDisplay.Items.Add = strFirstName.PadRight(20) & _
                       strSurname.PadRight(20)

strFirstName = "Tumelo"
strSurname = "Moeketsi"
lstDisplay.Items.Add = strFirstName.PadRight(20) & _
                       strSurname.PadRight(20)
strFirstName = "Daniel"
strSurname = "Nkosi"
lstDisplay.Items.Add = strFirstName.PadRight(20) & _
                       strSurname.PadRight(20)
```

The following will be displayed in the list box:

```
Sally               Jones
Mpho                Moeketsi
Daniel              Nkosi
```

1.8 COMPARE TWO STRINGS

Two strings can be compared to determine if they are equal or not equal by using the **Compare method**.

General format: *String*.Compare(firstString, secondString)

When the 2 strings are equal, the number 0 is returned.
When firstString > secondString, the number 1 is returned
When firstString < secondString, the number –1 is returned.

Example 1.8.1:

```
Dim intAnswer As Integer
intAnswer = String.Compare ("lion", "mouse")
```

The value in intAnswer will be -1 as the letter l is before the letter m in the alphabet (l is less than m).

Example 1.8.2:

```
Dim intAnswer As Integer
Dim strOne As String = "dog"
intAnswer = string.Compare (strOne, "dog")
```

The value in intAnswer will be 0 as they are equal.

Example 1.8.3:

```
Dim intAnswer As Integer
Dim strOne As String = "dog"
Dim strTwo As String = "cat"
intAnswer = string.Compare(strOne, strTwo)
```

The value in intAnswer will be 1 as the d in dog is positioned after c of cat in the alphabet.

1.9 RETURN A SUBSTRING (PORTION) OF A STRING

The **Substring method** is used to access one or more characters in a given string.

General format: *String*.Substring(beginIndex [, count])

The argument *beginIndex* is the index of the first character that you want to access in the string. The *count* argument specifies the number of characters that you want to access. *Count* is optional and if it is omitted, all characters as from the position specified by *beginIndex* will be returned.

Example 1.9.1:

```
Dim strWord1 As String = "Application"
Dim strWord2 As String

strWord2 = strWord1.Substring(5,2)
```

The value of strWord2 will be "ca"

Example 1.9.2:

```
Dim strWord1 As String = "Application"
Dim strWord2 As String

strWord2 = strWord1.Substring(5)
```

The value of strWord2 will now be "cation"

1.10 SEARCHING A STRING

1.10.1 STARTSWITH METHOD

Sometimes it might be necessary to test if a string begins with a specific character or set of characters. For instance, it might be necessary to test if a cell phone number starts with 083.

The StartsWith method determines whether a specific set of characters occurs at the beginning of a string.

General format: *String*.StartsWith(subString)

The answer of this method will be returned as a Boolean value and it can be tested in a selection statement.

Example 1.10.1.1

```
Dim strNumber As String = "0834254290"
If strNumber.StartsWith("083") Then
   lstDisplay.Items.Add("You are a MTN customer")
Else
   lstDisplay.Items.Add("You are not necessarily a MTN customer")
End If
```

1.10.2 ENDSWITH METHOD

The EndsWith method determines whether a specific set of characters occurs at the end of a string and it also returns a Boolean value as an answer.

General format: *String*.EndsWith(subString)

Example 1.10.2.1

```
Dim strNumber As String = "P.O.Box 211, Pretoria, 0001"
If strNumber.EndsWith("0001") Then
   lstDisplay.Items.Add("You live in Pretoria")
End If
```

1.10.3 CONTAINS METHOD

The **Contains method** searches a string value to determine whether it contains a specific sequence of characters or not and will return a TRUE or FALSE Boolean answer depending on the outcome of the search.

General format: *String*.Contains(subString)

Example 1.10.3.1:

```
Dim strNumber As String = "P.O.Box 211, Pretoria, 0001"
If strNumber.Contains("Pretoria") Then
   lstDisplay.Items.Add("You live in Pretoria")
End If
```

1.11 GENERAL EXAMPLES

1.11.1 Example 1

Problem:

The Big Bang Company needs to update and change the 8 department names and department codes in their store. The input data is read, validated, changed if needed and then the department names, old department codes and new department codes are displayed. However, if the department name is invalid, it must only be indicated as such and the old and new department code must not be displayed.

The name of a department may not be more than 12 characters.

The code consists of letters and numbers. The same number will not appear more than once in the code. A "D" must be inserted as the 1st character of the code and if the code contains the number 6 it must be replaced by the number 2.

Planning:

	Description	Type	Name of variable
Input:	department name	string	deptName
	department code	string	deptCode
Output:	department name	string	deptName
	old department code	string	deptCode
	new department code	string	newDeptCode

INPUT	PROCESSING	OUTPUT
deptName deptCode	Prompt for deptName, deptCode Verify input fields Modify if needed Display output	deptName deptCode newDeptCode

Algorithm:

VerifyDepartments

~This program verifies department names and changes department codes

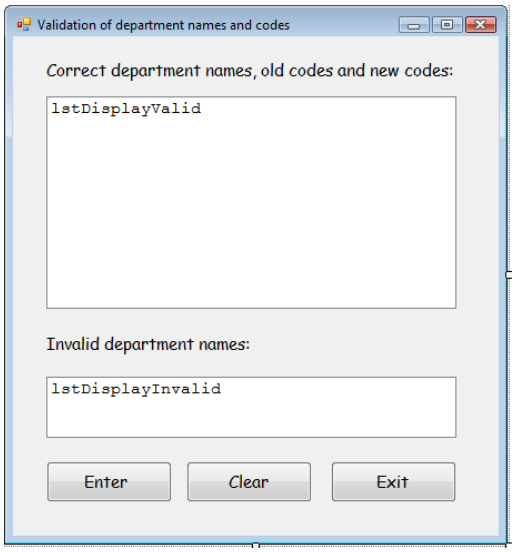
```
for count = 1 to 12
        display "Provide the name for department  ", count
        enter deptName
        if deptName.Length > 12
                display deptName      ~ As invalid
        else
                display deptName
                display "Provide the code for department ", count
                enter deptCode
                newDeptCode = deptCode.insert(0, "D")
                newDeptCode = newDeptCode.replace("6","2")
                display deptCode
                display newDeptCode
        endif
next count
```

Test the program using the following test data:

Department Name	Department Code	Department Name	Department Code
Sport	S589	Toys	TM0987
Children	AND654	Haberdashery	R3949B
Ladies	PU23	Linen and Crockery	GKJ48
Men	GR4678	Computer	CVB36

User Interface:

Two list boxes are used, one (lstDisplayValid) to display valid department names followed by the old department code and new department code, and the other (lstDisplayInvalid) to display invalid department names.



Code in the code editor:

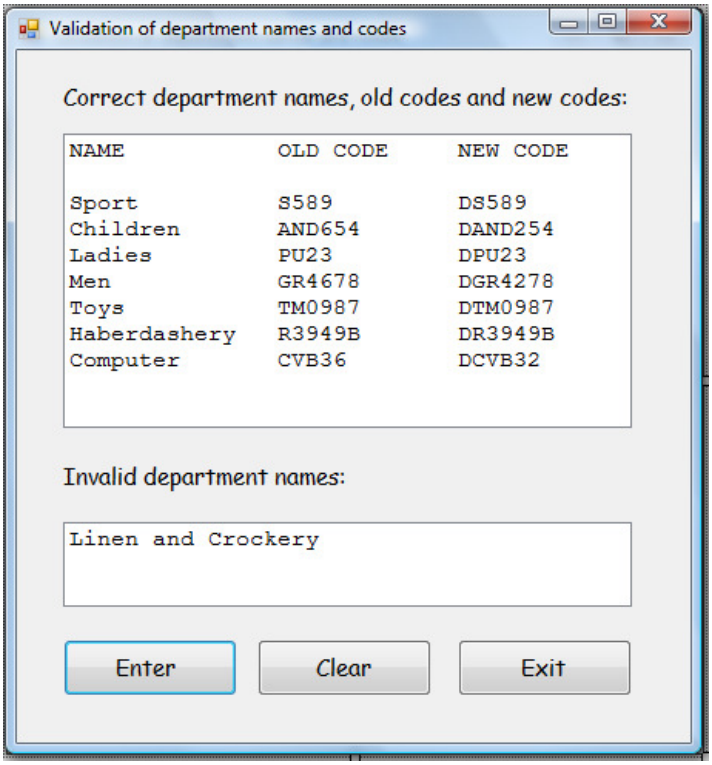
```
Public Class DeptNewCodes

Private Sub btnEnter_Click(. . .) Handles btnEnter.Click
   Dim strDeptName, strDeptCode, strNewDeptCode As String
   lstDisplayValid.Items.Add("NAME".PadRight(15) & _
                             "OLD CODE".PadRight(13) & "NEW CODE")
   lstDisplayValid.Items.Add(" ")
   For count As Integer = 1 To 8
       strDeptName = InputBox("Provide the name for department" & _
                              count.ToString)
       If strDeptName.Length > 12 Then
          lstDisplayInvalid.Items.Add(strDeptName)
       Else
          strDeptCode = InputBox("Provide the code for department " & _
                                 count.ToString)
          strNewDeptCode = strDeptCode.Insert(0, "D")
          strNewDeptCode = strNewDeptCode.Replace("6", "2")
          lstDisplayValid.Items.Add(strDeptName.PadRight(15) & _
                           strDeptCode.PadRight(13) & strNewDeptCode)
       End If
    Next
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   lstDisplayValid.Items.Clear()
   lstDisplayInvalid.Items.Clear()
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Output:



1.11.2 Example 2

Problem:

You are asked to read a sentence. If the sentence has an even number of characters, including spaces, display the sentence. If it has an odd number of characters, including spaces, include a hyphen as the 1st character and another one as the last character of the sentence. Show this changed sentence in a message box. Repeat the process until the characters xxx are entered.

Planning:

	Description	Type	Name of variable
Input:	original sentence	string	sentenceOne
Output:	changed sentence	string	sentenceTwo

INPUT	PROCESSING	OUTPUT
sentenceOne	Prompt for sentence Enter sentenceOne Get length of sentenceOne If odd, add hyphens Display sentenceTwo	sentenceTwo

Algorithm:

TestSentences

```
~This program changes sentences with an odd number of characters
        display "Enter a sentence"
        enter sentenceOne
        do while sentenceOne <> "xxx"
                numberChar = sentenceOne.length
                ~  the index of last character in the sentence is equal to the length - 1
                if numberChar MOD 2 = 0 then        ~ it contains even number of characters
                        sentenceTwo = sentenceOne
                else
                        sentenceTwo = sentenceOne.insert(0, "-" )
                        sentenceTwo = sentenceTwo.insert((numberChar+1), "-" )
                endif
                display sentenceTwo
                display "Enter a sentence, xxx to terminate"
                enter sentenceOne
        loop
end
```

Code in the code editor:

```
Public Class StringsEx2

Private Sub btnEnter_Click(. . .) Handles btnEnter.Click
   Dim strSentenceOne, strSentenceTwo As String
   Dim intNumberChar As Integer
   strSentenceOne = InputBox("Enter a sentence", _
                             "Enter xxx or press cancel to stop")
   Do While strSentenceOne <> "xxx" And strSentenceOne <> ""
      intNumberChar = strSentenceOne.Length
      If intNumberChar Mod 2 = 0 Then
         strSentenceTwo = strSentenceOne
      Else
         strSentenceTwo = strSentenceOne.Insert(0, "-")
         strSentenceTwo = strSentenceTwo.Insert((intNumberChar + 1), "-")
      End If
      lstDisplay.Items.Add(strSentenceTwo)
      strSentenceOne = InputBox("Enter a sentence", _
                                "Enter xxx or press cancel to stop")
   Loop
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   lstDisplay.Items.Clear()
End Sub

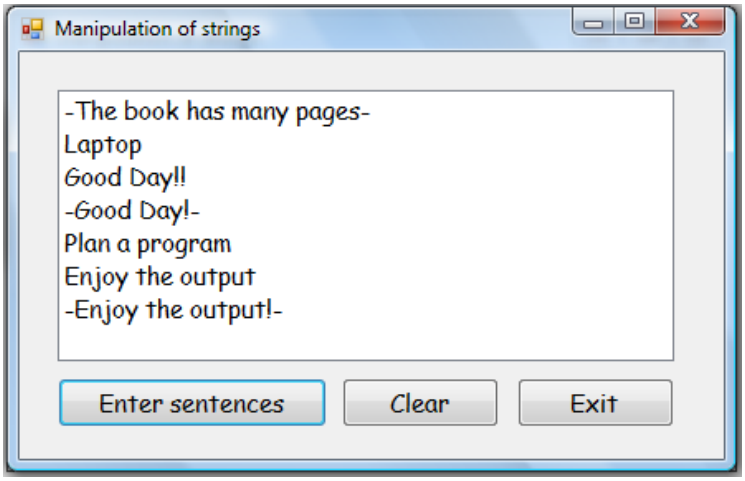
Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnExit.Click
   Close()
End Sub
End Class
```

Test the program using the following test data:

The book has many pages
Good day!!
Plan a program
Enjoy the output!

Laptop
Good Day!
Enjoy the output

Output:





Exercises

Study the following declarations and then answer the questions that follow. Use the original declaration in each question unless stated otherwise:

```
Dim strSentenceOne As String = "Today is a lovely day"
Dim strSentenceTwo As String
Dim strSentenceThree As String = "How many days in a week? "
Dim intNumber As Integer
```

a. strSentenceTwo = strSentenceOne.Substring(9)

The value of strSentenceTwo is _____

b. strSentenceTwo = strSentenceOne.Replace("Today","Tonight")
strSentenceTwo = strSentenceTwo.Replace("day","night")

The value of strSentenceTwo is _____

c. intNumber = strSentenceOne.Length
strSentenceTwo = strSentenceOne.Remove((number-10),7)

The value of strSentenceTwo is _____

d. Use the values of strSentenceOne and strSentenceTwo obtained in question c and process the following statement:

intNumber = String.Compare (strSentenceOne, strSentenceTwo)

The value of intNumber is _____

e. strSentenceTwo = strSentenceThree.Remove(0,3)
strSentenceTwo = strSentenceTwo.Insert(0, "Seven")
strSentenceTwo = strSentenceTwo.Replace("?"," ")

The value of sentenceTwo is _____



1. Solve the following problem by doing the planning and writing a VB.NET program.

A word is entered into the program and stored in the variable *firstWord*. More words are entered one by one and the program will terminate as soon as a word is entered that has the same value as *firstWord.* Count and display on a label how many words were entered before a word that is equal to *firstWord* is entered.

Test your program using the following data:

Information
Computer
Intelligence
Communication
Information

(The count displayed on the label should be 4)

2. Solve the following problem by doing the planning and writing a VB.NET program.

According to the legend, the island of Manhattan was purchased by the Dutch from Manhattan Indians in 1626 for $24. Assuming that this money was invested in a Dutch bank paying 5% simple interest per year, calculate and display how much money the Indians would have at the end of each 20 year period starting from 1626 up to the current year. Each output line should reflect the year and the amount.

Display these answers in a list box and make use of the PadLeft and PadRight methods to align output.

3. Solve the following problem by doing the planning and writing a VB.NET program.

You have received a list of 10 names of people that must be entered using a loop. The name only contains a Mr or Ms to indicate the gender, followed by a space, followed by an initial, followed by a space and lastly the surname e.g.

Mr M Jones
Ms P Flower

Your program must use string manipulation methods to remove the Mr/Ms and insert a first name of your choice, according to the gender and display these new names in a listbox e.g.

Martin Jones
Petunia Flower

CHAPTER 2
FUNCTION PROCEDURES AND SUB PROCEDURES

2.1 INTRODUCTION

Using function procedures and sub procedures is a way to make computer coding more effective by splitting a big complex algorithm into smaller modules.

Lines of code that perform a specific task that is repeated often, would typically be coded as a separate procedure. The procedure is therefore written once and can then be called whenever this task needs to be done. Large and complex applications can also be broken down into small manageable tasks. It is often the case that more than one programmer simultaneously work on the same application, each on a different sub procedure.

A sub procedure and a function procedure will perform a task but a function procedure (or just called a function) will also always return a value that has been determined or calculated within the function.

OUTCOMES

When you have studied this chapter you should be able to know and understand the following:

- Purpose and use of value parameters
- Purpose and use of reference parameters
- Use of function procedures
    - Write a call statement to a function
    - Write the corresponding function header and statements to return an answer
    - Write functions with and without parameters
- Use of sub procedures
    - Write a call statement to a sub procedure
    - Write the corresponding sub procedure header and statements in the sub procedure
    - Write sub procedures with or without parameters

2.2 PARAMETERS

Before we can study function procedures and sub procedures we need to understand what parameters are.

A parameter is data that is sent to a function or a sub procedure to be used in the task it has to do.

There are 2 different types of parameters:

- Value parameters
- Reference parameters

2.2.1 Value parameters

Value parameters can be explained by using the following example used in everyday life:

Joseph has a page on which he wrote some numbers.
He makes a copy of the page and sends it to Solly. (Joseph sends value parameters to Solly who receives the value parameters. Solly could be a function or a sub procedure).

Solly may do anything he wishes with the parameters as Joseph is not interested in this copy anymore. He has the original.

Solly may use these parameters to calculate a total. He may keep this total for his own use or he can return it to Joseph.

If he keeps it, the total could be printed or it can be used in other calculations.

If he returns the value of the total to Joseph, Solly is a function and not a sub procedure because a function always returns an answer.

Joseph will now have his original copy as well as a total that does not exist on his original copy.

Another everyday example is that Sam may tell Henry how much money he has in his bank account (sending a copy i.e. a value parameter) to function Henry. Function Henry will now be able to calculate how much money the two of them have in total (but he cannot change the amount in Sam's account).

In the generic planning of the programs to follow we will use the prefix **val** for a value parameter.

2.2.2 Reference parameters

Once more, an everyday example can be used to explain reference parameters:

Josephine receives a post box number and the key of this postal box at the post office. She goes to the post office and opens the box to find a letter in the box. She can now put something else in the box that the owner will collect. She can also receive more than one post box number (address) and key to return different letters to the owners.

In other words: Two owners of post boxes give their keys and post box numbers (2 reference parameters) to sub procedure Josephine. This means that Josephine knows the addresses of the post boxes and she has access to the post boxes. These post boxes represent different areas in memory. Sub procedure Josephine may now remove the letter, put another letter in the post box or just write something else on the letter. However, bear in mind that while the instructions apply; , changes are issued within sub procedure Josephine. These changes are taking place within the post boxes.

Now we can look at the banking example: Sam gives sub procedure Henry access to his banking account by giving him the number and pin code. Procedure Henry can withdraw an

amount or deposit an amount of money in Sam's account. These changes will take place in Sam's account by all the instructions, which were given by sub procedure Henry.

In the generic planning of the programs to follow we will use the prefix **ref** for a reference parameter.

2.3 FUNCTION PROCEDURES

A function procedure (we will refer to it as a function) consists of a number of statements to execute a single task. It may receive some value parameters and will always return a value. It is possible that it may receive no parameters.

2.3.1 The function call

A function is called from another part of the program and after the function has been processed, the control will return to the calling statement. In the function call, the list of parameters is called the argument list.

The calling statement may be of different types (assignment statement, if-statement, display statement) as can be seen in the following examples. In each case the actual call consist of the name of the function followed by an argument list in brackets. There may be no arguments inside the brackets:

1. Send 2 values to a function named *CalcSum* that will calculate the sum of these two numbers.

   Call the function:
   sum = CalcSum (number1, number2)
   The answer will be stored in *sum*. *CalcSum* is the name of the function while *number1* and *number2* are the arguments (2 value parameters).

2. Use the same example, but do not store the answer in *sum* rather display the sum of the 2 numbers:

   display "The value of the sum is: ", CalcSum(number1, number2)

   The function *CalcSum* is called while sending the 2 value parameters *(number1* and *number2)* to the function. The sum is returned to the calling statement and displayed on the screen of the computer.

3. Another type of example is the following:

   ```
   if CalcSum(number1, number2) > 40
         display "The sum is greater than 40"
   endif
   ```

   The function *CalcSum* is called passing the 2 value parameters to the function to calculate the sum of the 2 numbers. The sum is returned to the calling statement and then used in the if-statement.

2.3.2 The function

The first line of a function, called the function header contains the word function, a function name and a list of parameters. Once again, the parameter list may be empty.

Format: Function NameOfFunction (parameter list)

The last statement of the function to be processed will be a return statement (exit the function) that sends the answer of the function to the calling statement. There may be more than one return-statement in a function – e.g. depending on the outcome of an if-statement, one of various different values could be returned to the calling module, but all must be of the same data type.

Format: return value

The name of the function is associated with the data type of the value that is returned to the calling module. For this reason, a function may return only one value.

Example:

Write the calling statement to call a function that calculates the sum of 2 numbers and store the answer in the variable *sum.*

Calling statement:

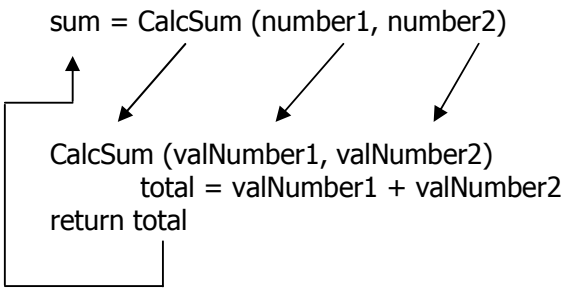
```
sum = CalcSum (number1, number2)
```

The function to calculate the sum of two numbers will be as follows:

```
Function CalcSum (valNumber1, valNumber2)
        total = valNumber1 + valNumber2
return total
```

Note the following:

- The function name used in the calling statement and the function name in the function header must always be the same.
- The names of the arguments and parameters need not be the same.
- The order of the variables in the argument list of the calling statement and the parameters in the function header must always be the same.
- The name of the value returned from the function need not be the same as in the calling statement.
- The arguments are only the names of the applicable variables in the calling module. On the other hand, the parameters in the header must indicate whether the parameter is a value parameter or a reference parameter. I.e. whether a copy or an address has been sent to it.



The function can also be written as follows where the variable *total* is replaced by the expression valNumber1 + valNumber2.

```
Function CalcSum (valNumber1, valNumber2)
return valNumber1 + valNumber2
```

2.3.3 Format of a function call and function header in Visual Basic.NET

The function call is exactly the same as in pseudocode:

```
Statement NameOfFunction (Argument list)
```

The syntax of the function procedure is as follows:

```
Private Function NameOfFunction ([parameterlist]) As data type
    [Statement(s) in the function]
    Return Value of the answer
End Function
```

The procedure header contains the name of the function, followed by an optional parameter list and the data type of the answer that will be returned to the function call. Recall that the name of the function is associated with the value that it returns to the calling module.

A function may contain one or more statements. However, it is also possible that there could be no statements as the calculation of the answer could appear in the return statement. The return statement may contain a variable name or an expression. The data type of this variable or expression must correspond to the data type at the end of the procedure header. This statement will end the function and return to the calling statement. Every function is terminated by End Function (the procedure footer). VB.NET will automatically insert the procedure footer during the coding process when a valid procedure header statement has been entered.

2.3.4 Examples calling a function and using a function

Do the planning and solve the problem by writing a VB.NET program for each of the following:

2.3.4.1 EXAMPLE 1

Reginald went to a shop to buy a number of fruit bars. The number bought and the price of a fruit bar is entered in a main procedure. Send these numbers to a function to calculate the amount due. The amount due must include 14% VAT. Return this value to the main procedure and display the amount on the screen.

Planning:

	Description	Type	Name of variable
Main procedure (ShoppingProgram):			
Input:	number of items	integer	numItems
	price of one item	real	price
Output:	amount due	real	amtDue
Function (CalcAmtDue):			
Input:	copy of number of items	integer	valNum
	copy of price of one item	real	valPrice
Output:	amount due	real	- Not used -

Algorithm:

Function CalcAmtDue (valNum, valPri
Return (valNum * valPrice * 1.14)

ShoppingProgram

~ The input data will be entered in the main procedure.
~ The main procedure will then call a function to calculate the amount due
~ The amount due will be displayed in the main procedure.

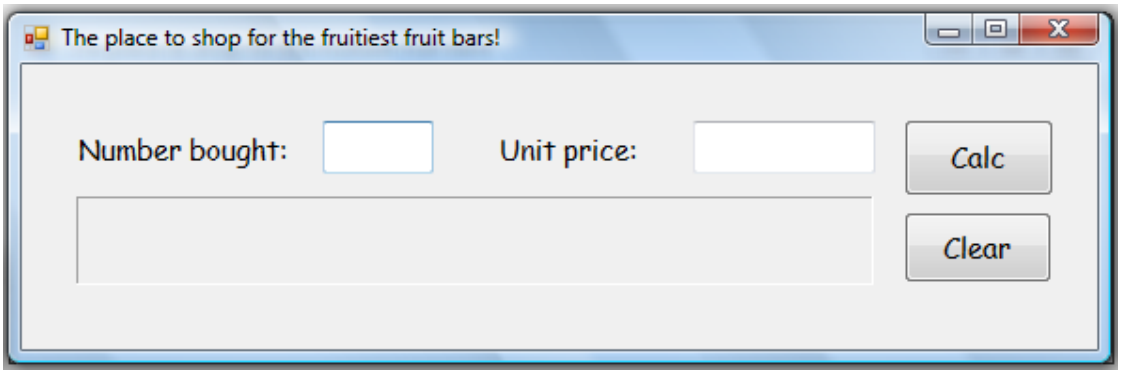
```
display "Enter the number of fruit bars bought"
enter numItems
display "Enter the price of a fruit bar"
enter price
amtDue = CalcAmtDue (numItems, price)    ~  function call
display "The amount due is R", amtDue
end
```

Note:

- This example illustrates that the function and the main program are two completely separate entities.
- The main program and the function communicate by using parameters.
- If the function does not receive the number and the price, it will not be able to calculate the correct answer.
- If the amount due is not returned to the main procedure, the main procedure will not have the correct value to display.
- The parameters in the argument list of call statement in the main procedure must be in the same order as the parameters in the parameter list of the function header.
- In most programming languages, the functions and sub procedures appear at the beginning of the code. Although this is not a necessity in VB.NET, all functions and sub procedures will appear at the beginning of the code in the code editor as well as in the algorithm.

VB.NET Application:

User Interface:



Code in the code editor:

```
Public Class Shopping

' Procedure Header
' It contains two value parameters and the data type of the returning value
' is decimal
Private Function CalcAmtDue(ByVal intNum As Integer, _
                            ByVal decPrice As Decimal) As Decimal
   ' The function contains no additional statements
   ' The calculation is done in the return statement
   Return Convert.ToDecimal(intNum) * decPrice * 1.14D
End Function

Private Sub btnCalc_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCalc.Click
   Dim intNoItems As Integer
   Dim decPrice, decAmtDue As Decimal
   Integer.TryParse(txtNoItems.text, intNoItems)
   Decimal.TryParse(txtPrice.text, decPrice)
   decAmtDue = CalcAmtDue(intNoItems, decPrice)
   lblAnswer.Text = "The amount due = R" & decAmtDue.ToString("N2")
End Sub

Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnClear.Click
   txtNoItems.Text = ""
   txtPrice.Text = ""
   lblAnswer.Text = ""
   txtNoItems.Focus()
End Sub

End Class
```

Example of output:



2.3.4.2 EXAMPLE 2

Write a program to enter three test marks for a student. A function is called to calculate the average mark. Another function is used to indicate if the average mark is a pass mark (>=50) or a fail mark. The average as well as the result must be displayed on the screen. All input and output must be done in the main procedure.

Planning:

	Description	Type	Name of variable
Main procedure (ShowResults):			
Input:	student name	string	studentName
	test mark 1	integer	testMark1
	test mark 2	integer	testMark2
	test mark 3	integer	testMark3
Output:	average mark	real	average
	result	string	- not used -
Function (CalcAve):			
Input:	copy of test mark 1	integer	valTest1
	copy of test mark 2	integer	valTest2
	copy of test mark 3	integer	valTest3
Output:	average mark	real	- not used -
Function (DetermineResults):			
Input:	copy of average mark	integer	valAve
Output:	result message	string	message

Algorithm:

```
Function CalcAve (valTest1, valTest2, valTest3)          ~    function
~       Calculate the average of the three tests
return (valTest1 + valTest2 + valTest3) / 3

Function DetermineResult (valAve)                        ~    function
~       Produce a message to express the outcome
        message = "pass"
        if valAve < 50 then
                message = "fail"
        endif
return message


ShowResults                                              ~    main procedure
~       This program determines the result of the student's performance
        display "Enter the student name"
        enter studentName
        display "Enter the first test mark"
        enter testMark1
        display "Enter the second test mark"
        enter testMark2
        display "Enter the third test mark"
        enter testMark3

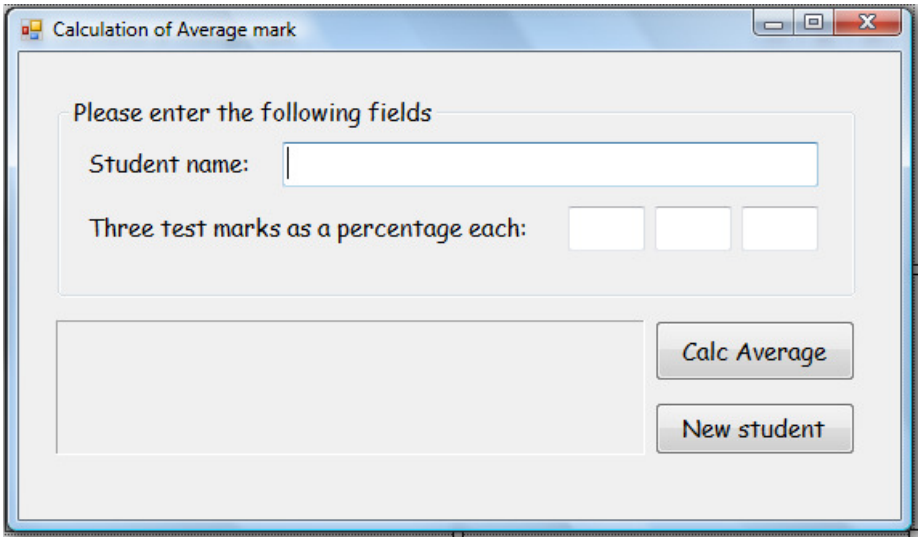
        ~       call the function to calculate the average
        average = CalcAve (testMark1, testMark2, testMark3)
        display "The average mark for ", studentName , " is  ", average

        ~       call the function to determine the result
        display "This student will  ", DetermineResults(average)

end
```

VB.NET Application:

User Interface:



Code in the code editor:

```
Public Class ShowResults

Private Function CalcAve(ByVal intTest1 As Integer, _
   ByVal intTest2 As Integer,ByVal intTest3 As Integer) As Decimal
   Return Convert.ToDecimal((intTest1 + intTest2 + intTest3) / 3)
End Function

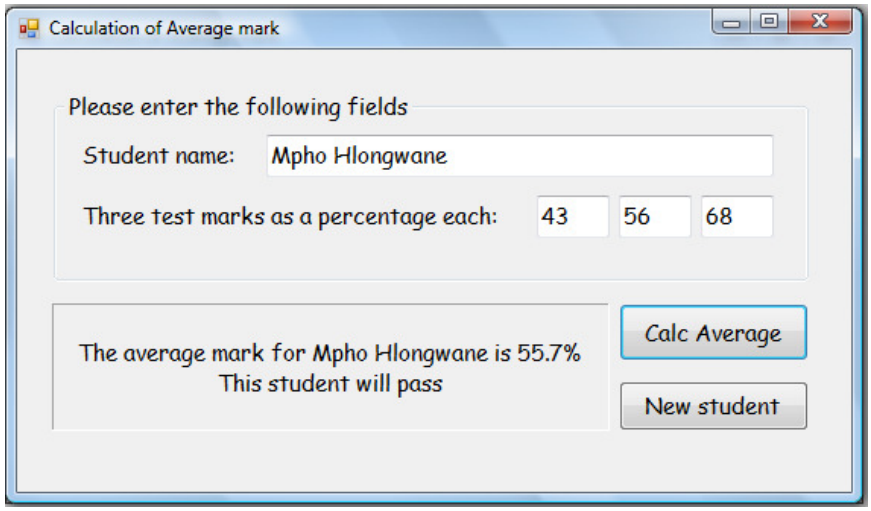
Private Function DetermineResults(ByVal decAve As Decimal) As String
   Dim strMessage As String = "pass"
   If decAve < 50 Then
      strMessage = "fail"
   End If
   Return strMessage
End Function

Private Sub btnCalc_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCalc.Click
   Dim strStudentName As String
   Dim intTestMark1, intTestMark2, intTestMark3 As Integer
   Dim decAverage As Decimal
   strStudentName = txtStudentName.Text
   Integer.TryParse(txtTest1.Text, intTestMark1)
   Integer.TryParse(txtTest2.Text, intTestMark2)
   Integer.TryParse(txtTest3.Text, intTestMark3)
   decAverage = CalcAve(intTestMark1, intTestMark2, intTestMark3)
   lblAnswer.Text = "The average mark for " & strStudentName & " is " & _
                 decAverage.ToString("N1") & "%" & ControlChars.NewLine & _
                 "This student will " & DetermineResults(decAverage)
End Sub

Private Sub btnNewStudent_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNewStudent.Click
   txtStudentName.Text = ""
   txtTest1.Text = ""
   txtTest2.Text = ""
   txtTest3.Text = ""
   lblAnswer.Text = ""
   txtStudentName.Focus()
End Sub

End Class
```

Example of output:





Questions

Identify and correct all errors in the following function call and corresponding function header.

decPay = CalcPay(intHours, decTariff)

Private Function Pay(ByRef decTariff As Decimal, ByVal decHours As Decimal)



Exercises

Study each of the following function calls and then write the complete function procedure in VB.NET to calculate and return the answer.

1. intBestMark = DetermineBest(intTest1, intTest2, intTest3)

   The function must determine and return the **best** of the 3 test marks.

2. decAverage = CalcAverage(intTest1, intTest2, intTest3)

   The function must calculate and return the **average** of the 3 test marks.

3. decPay = CalcPay(intHours, decTariff)

   The function must calculate the pay for the person by multiplying the hours by the tariff per hour. If the hours are more than 40, the person receives 1½ times the tariff for all hours more than 40.

4. decNetMonthSal = CalcSal(decGrossAnnSal, decTaxPercentage)

   The function must calculate and return the net monthly salary for an employee if the annual gross salary is received as well as the percentage tax that the person must pay. Deduct the tax amount from the monthly gross salary in order to calculate the monthly net salary.



Practical

Plan the solution and then solve the problem by writing a VB.NET program. All input and output must be done in the main procedure.

1. Enter the radius of a circle and then call a function to calculate the circumference and another function to calculate the area of the circle. Display the calculated values on the screen.

2. Elize buys a number of boxes of smarties. You are asked to enter the number and the unit price of a box of smarties. The amount due is calculated as the number bought times the price. However, a customer may receive discount, depending on the number of boxes bought as can be seen in the table below. All calculations for the amount due will be calculated in a function. The function will receive the number bought and unit price. Display the amount due in the main procedure.

Number of boxes	Discount percentage
1 - 7	No discount
8 - 20	2.5%
21 and more	4.75%

3. You are a bookkeeper for a company and you have to split all amounts you paid to suppliers into the basic amount and the VAT. The VAT rate of 14% applies. Example: If the amount is R114, the basic amount will be R100 and the VAT amount will be R14. Enter the final amount, calculate the VAT in a function and the basic amount in another function. Display the basic amount and the VAT on the screen. Repeat this process for 4 different amounts.

2.4 SUB PROCEDURES

A sub procedure is also used to solve a specific part of the problem. It has the following properties:

- It may receive value parameters to use in the sub procedure.
- It may receive reference parameters.
- It does not return a value.
- It is able to manipulate data outside the sub procedure at specific addresses as given in the reference parameters i.e. it can change more than one value.

2.4.1 The Sub Procedure call

The syntax of the sub procedure call used in the planning is the following:

```
Call  NameOfSubProcedure (argument list)
```

Study the following sub procedure call:

```
Call Calculations (number1, number2, sum, product)
```

Calculations is the name of the sub procedure.
number1 and *number2* are value parameters containing numeric values. These values may not be changed by the sub procedure.
sum and *product* are two reference parameters. These values may be changed by the sub procedure.

Actually, we are going to use the sub procedure to calculate **two** values i.e. the sum and the product of the two numbers.

2.4.2 The Sub Procedure

The syntax of the header line of a sub procedure used in the planning is the following:

```
Sub NameOfSubProcedure (parameter list)
```

Let us write the sub procedure for the sub procedure call in the previous paragraph:

```
Sub Calculations (valno1, valno2, refsum, refproduct)
```

Calculations is the name of the sub procedure (the same name as in the sub procedure call)
valno1 and valno*2* are value parameters containing the numeric values sent from the sub procedure call.
refsum and ref*product* are two reference parameters that contain the addresses of the sum and the product.

The sub procedure:

```
Sub Calculations (valno1, valno2, refsum, refproduct)
~       calculate the sum and the product of 2 numbers
        refsum = valno1 + valno2
        refproduct = valno1 * valno2
End Sub
```

The sub procedure receives "copies" of the values of *number1* and *number2* using parameters.
refsum and *refproduct* contain the addresses of the area in memory where the sub procedure must store the result of the calculation. The calling module will also refer to these addressed and will therefore be able to get hold of the answers.

The line *End Sub* terminates the sub procedure.

A function or sub procedure can be called from a main procedure or from any other function or sub procedure as can be seen in the examples.

2.4.3 Format of an independent sub procedure call and a sub procedure header in Visual Basic.NET

Sub procedures are no new topic and you have been using it since level 2. Visual Basic contains two types of sub procedures - Event procedures and independent sub procedures. An **event procedure** is a sub procedure associated with a specific object and event. E.g. a button click event. The computer will automatically process the event procedure when the event occurs. An independent sub procedure is a collection of statements that can be invoked from one or more places in an application. It will be processed when it is called or invoked from code at a specific point in the application.

The independent sub procedure call is exactly the same as in pseudocode:

```
Call NameOfSubProcedure (Argument list)
```

The syntax of the sub procedure is as follows:

```
Private Sub NameOfSubProcedure ([parameterlist])
    Statement(s) in the sub procedure
End Sub
```

The procedure header contains the name of the sub procedure, followed by an optional parameter list. A sub procedure must contain one or more statements and will contain no return statement. Every sub procedure is terminated by End Sub (the procedure footer). VB.NET will automatically insert the procedure footer when a valid sub procedure header has been entered.

2.4.4 Examples calling an independent sub procedure and using a sub procedure

Do the planning and solve the problem by writing a VB.NET program for each of the following:

2.4.4.1 EXAMPLE 1

Danny invested an amount that must be entered at the beginning of the program at the SAVE-A-LOT BANK. The monthly interest rate is also entered. The program must calculate and display the amount of interest earned as well as the balance at the end of every month for the next 15 months. At the end of the program the total amount of interest must be displayed. Use functions and sub procedures where possible.
Remember when you test your program to use a monthly interest rate e.g. an annual interest rate of 10% = 0.83% per month.

Planning:

	Description	Type	Name of variable
Main procedure (EarnInterest):			
Input:	amount	real	amount
	interest rate	real	rate
Output:	amount	real	amount
	monthly interest	real	monInt
	total interest	real	totInterest
Sub procedure (CalcNewValues):			
Parameters:	copy of interest rate	real	valRate
	address of amount	real	refAmt
	address of interest rate	real	refMonthInterest
Sub procedure (AccSum):			
Parameters:	monthly interest	real	valMInterest
	address of total interest amount	real	refTInterest
Sub procedure (DisplayResults)			
Parameters:	copy of month number	integer	valMonth
	copy of amount	real	valAmt
	copy of monthly interest amount	real	valMInt

Algorithm:

```
Sub DisplaytResults (valAmt, valMInt)                    ~ display monthly values
        display valMonth, "       ", valAmt, "     ", valMInt
End Sub

Sub CalcNewValues (valRate, refAmt, refMonthInterest)
        refMonthInterest = refAmt * valRate / 100        ~ calculate interest
        refAmt = refAmt + refMonthInterest               ~ increase amount
End Sub

Sub AccSum (valMInterest, refTInterest)
        refTInterest = refTInterest + valMInterest       ~ accumulate interest total
End Sub
```

```
EarnInterest
~       This program deals with the investment of money
        totInterest = 0
        monInt = 0
        display "Month          Amount          Interest earned"      ~ heading

        display "Provide the amount you want to invest"
        enter amount
        display "Enter the monthly interest rate"
        enter rate

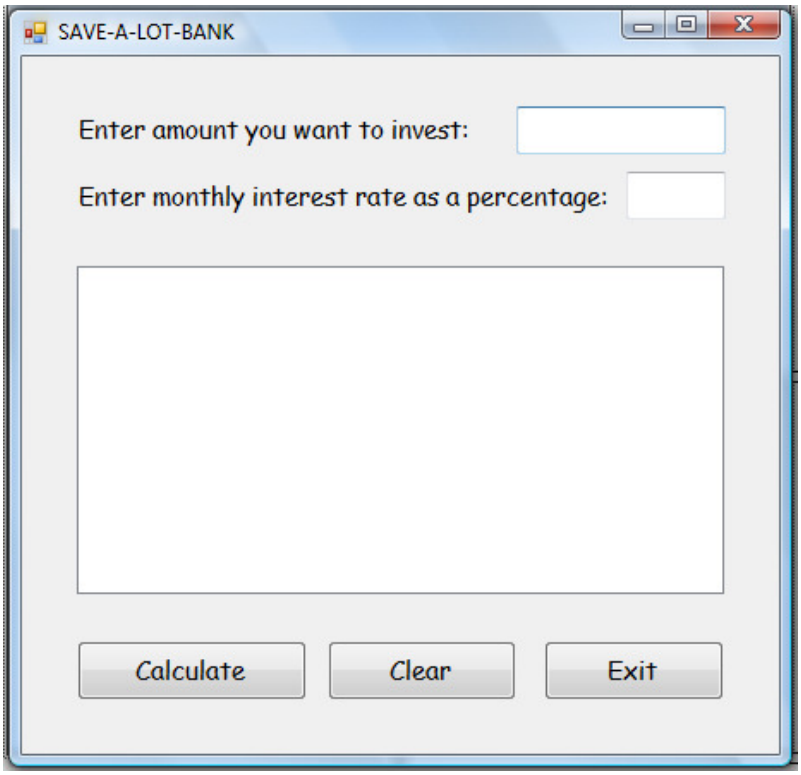
        call DisplayResults (0, amount, monInt)                       ~  display initial
values

~       repeat 15 times to obtain final results

        for x = 1 to 15
                call CalcNewValues (rate, amount, monInt)
                call DisplayResults (x, amount, monInt)
                call AccSum (monInt, totInterest)
        next x
        display "The total amount of interest earned is R", totInterest
end
```

VB.NET Application:

User Interface:



Take note:
In this example the PadLeft and PadRight methods that have been discussed in chapter 1 will be used to align the columns in the list box.

Code in the code editor:

```
Public Class EarnInterest

Private Sub DisplayResults(ByVal intMonth As Integer, _
            ByVal decAmt As Decimal, ByVal decMInt As Decimal)
   lstDisplay.Items.Add(intMonth.ToString.PadRight(5) & _
                        decAmt.ToString("N2").PadLeft(13) & _
                        decMInt.ToString("N2").PadLeft(20))
End Sub

Private Sub CalcNewValues(ByVal decRate As Decimal, _
            ByRef decAmt As Decimal, ByRef decMonthInterest As Decimal)
   Dim decInt As Decimal
   decInt = decRate / 100D
   decMonthInterest = decAmt * decInt
   decAmt = decAmt + decMonthInterest
End Sub

Private Sub AccSum(ByVal decMInterest As Decimal, _
                   ByRef decTInterest As Decimal)
   decTInterest += decMInterest
End Sub

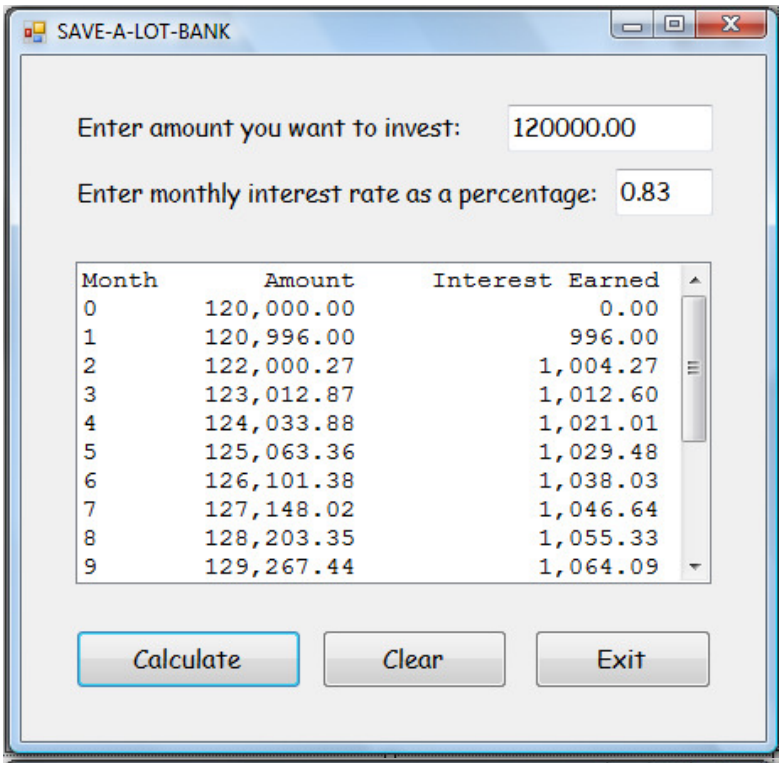
Private Sub btnCalc_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCalc.Click
   Dim decTotInterest As Decimal = 0
   Dim decMonInt As Decimal = 0
   Dim decAmount As Decimal
   Dim decRate As Decimal
   lstDisplay.Items.Add("Month" & "Amount".PadLeft(13) & _
                        "Interest Earned".PadLeft(20))
   Decimal.TryParse(txtAmount.Text, decAmount)
   Decimal.TryParse(txtRate.Text, decRate)
   Call DisplayResults(0, decAmount, decMonInt)
   For x As Integer = 1 To 15
       Call CalcNewValues(decRate, decAmount, decMonInt)
       Call DisplayResults(x, decAmount, decMonInt)
       Call AccSum(decMonInt, decTotInterest)
   Next
   lstDisplay.Items.Add(" ")
   lstDisplay.Items.Add("Total amt of interest earned " & _
                        decTotInterest.ToString("N2"))
End Sub

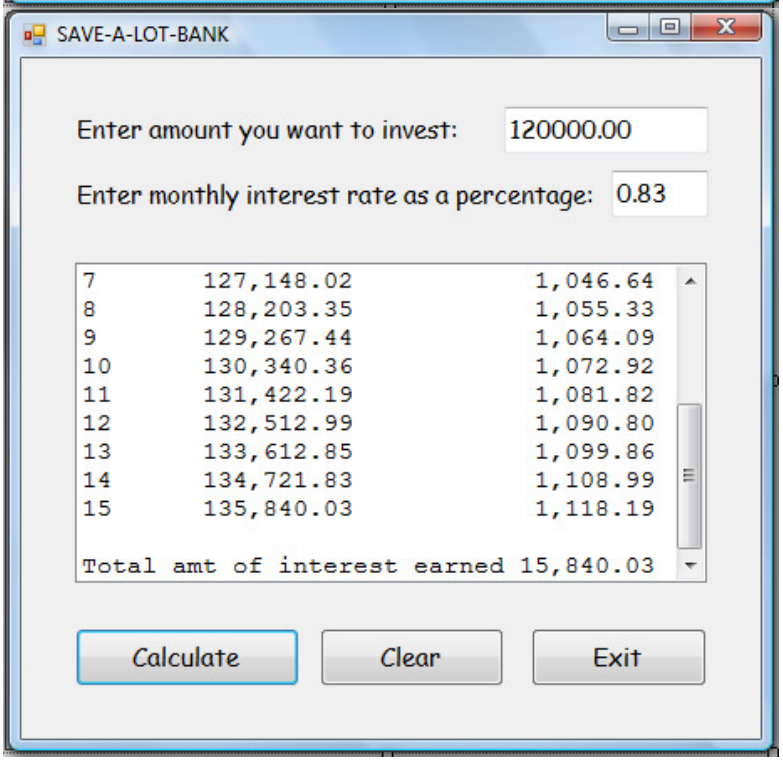
Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnClear.Click
   txtAmount.Text = ""
   txtRate.Text = ""
   lstDisplay.Items.Clear()
   txtAmount.Focus()
End Sub
```

```
Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnExit.Click
   Close()
End Sub

End Class
```

Example of output:





2.4.4.2 EXAMPLE 2

Do the planning and write a VB.NET program to calculate the final mark a student obtained for Programming 3. This final mark is calculated by using the marks of different assessments and their particular weights:

Assessment	Weight
Test 1	15%
Test 2	20%
The best mark of 2 class tests	15%
Examination	50%

Code a function to determine and return the mark of the highest class test.
Use a sub procedure to calculate the final mark and to determine whether it is a pass or fail (message) and another sub procedure to display the mark and the message result.

Planning:

	Description	Type	Name of variable
Main program (CalcFinalMark):			
Input:	mark for test 1	integer	test1
	mark for test 2	integer	test2
	mark class test 1	integer	clTest1
	mark class test 2	integer	clTest2
	exam mark	integer	exam
Output:	none		
Function (CalcClMark)			
Input:			
	copy of class test 1 mark	integer	valClMark1
	copy of class test 2 mark	integer	valClMark2
Output:			
	Best class test mark	integer	bestMark
Sub procedure (CalcValues):			
Parameters:	copy of test 1	integer	valTest1
	copy of test 2	integer	valTest2
	copy of highest class test	integer	valClassTest
	copy of exam mark	integer	valExam
Sub procedure (DisplayResults)			
Parameters:	copy of final mark	integer	valFinalMark
	copy of message of result	string	valMessage

Algorithm:

```
Function CalcClMark (valClMark1, valClMark2)                    ~ function
        if valClMark1 > valClMark2
                bestMark = valClMark1
        else
                bestMark = valClMark2
        endif
return bestMark

Sub DisplayResults (valFinalMark, valMessage)                   ~  sub procedure
        display "The final mark of the student is ", valFinalMark
        display "The result is ", valMessage
End Sub

Sub CalcValues (valTest1, valTest2,valClassTest,valExam)
                                                                ~  sub procedure
        Final = valTest1 * 0.15 + valTest2 * 0.2 + valClassTest * 0.15
                        + valExam * 0.5)
        in final >= 50
                message = "Pass"
        else
                message = "fail"
        endif
        Call DisplayResults (final, message)
End Sub

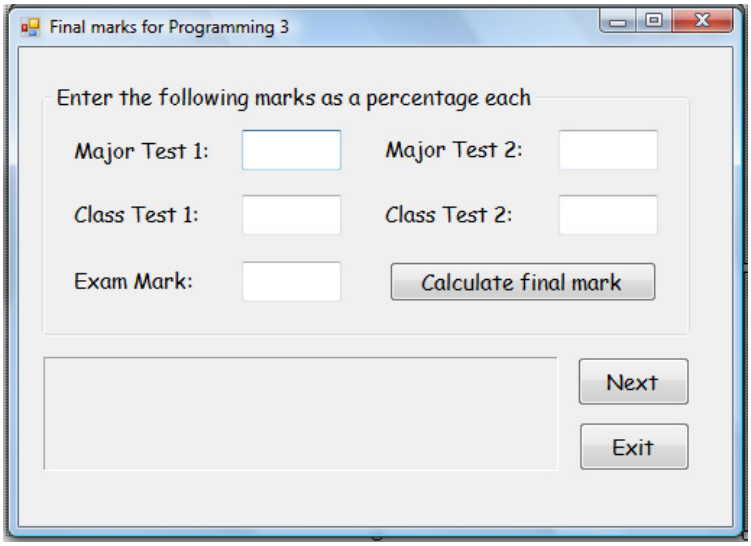
CalcFinalMark                                                   ~  Main program
        display "provide the mark for test 1"
        enter test1
        display "provide the mark for test 2"
        enter test2
        display "provide the mark for class test 1"
        enter clTest1
        display "provide the mark for test 1"
        enter clTest2
        display "provide the exam mark"
        enter exam
        highestClassTest = CalcClMark (clTest1, clTest2)        ~ function call
        call CalcValues (test1, test2, highestClassTest, Exam)  ~ call sub procedure
end
```

Take note:

As mentioned earlier, a function or sub procedure can be called from a main procedure or from any other function or sub procedure. In this program a function has been used to determine the highest class test and a sub procedure is used to determine the result.

VB.NET Application:

User Interface:



Code in the code editor:

```
Public Class CalcFinalMark

Private Function CalcClMark(ByVal intClMark1 As Integer, _
                            ByVal intClMark2 As Integer) As Integer
   Dim intBestMark As Integer
   If intClMark1 > intClMark2 Then
      intBestMark = intClMark1
   Else
      intBestMark = intClMark2
   End If
   Return intBestMark
End Function

Private Sub DisplayResults(ByVal decFinalMark As Decimal, _
                           ByVal strMessage As String)
   lblAnswer.Text = "The final mark of the student is " & _
                    decFinalMark.ToString("N1") & "%" & _
                    ControlChars.NewLine & "The result is a " & strMessage
End Sub

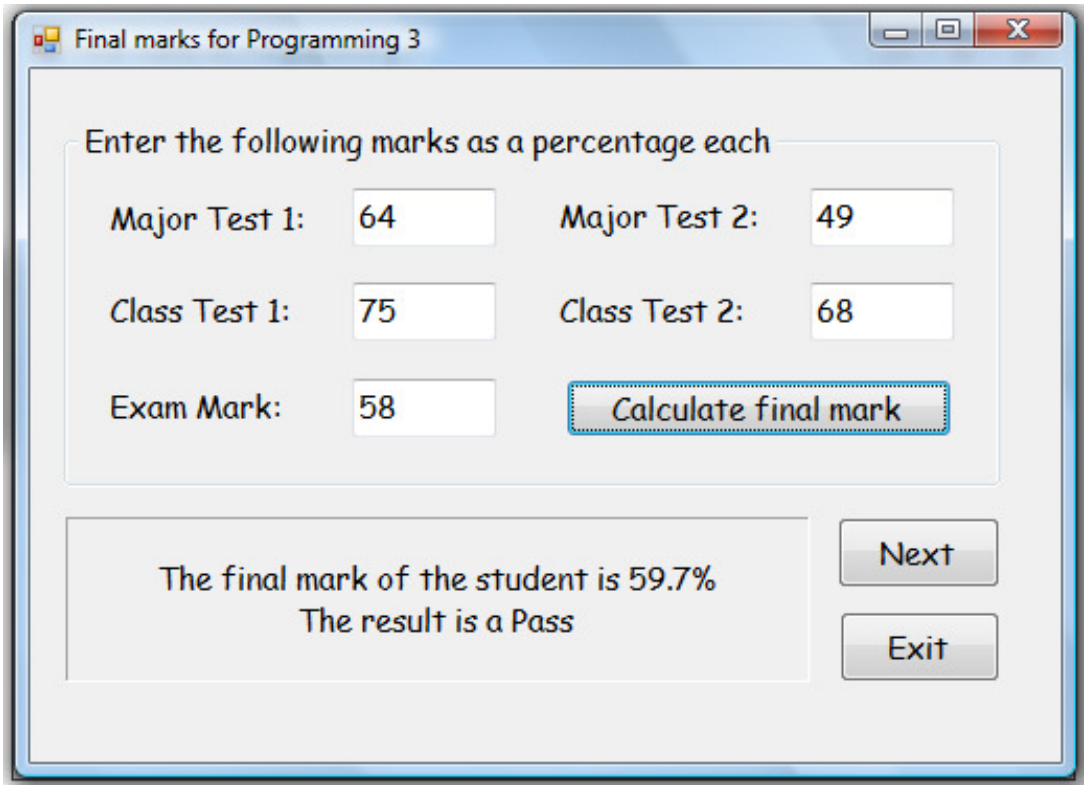
Private Sub CalcValues(ByVal intTest1 As Integer, _
            ByVal intTest2 As Integer, ByVal intClassTest As Integer, _
            ByVal intExam As Integer)
   Dim decFinal As Decimal
   Dim strMessage As String
   decFinal = Convert.ToDecimal(intTest1 * 0.15 + intTest2 * 0.2 _
              + intClassTest * 0.15 + intExam * 0.5)
   If decFinal >= 50 Then
      strMessage = "Pass"
   Else
      strMessage = "Fail"
   End If
   Call DisplayResults(decFinal, strMessage)
End Sub
```

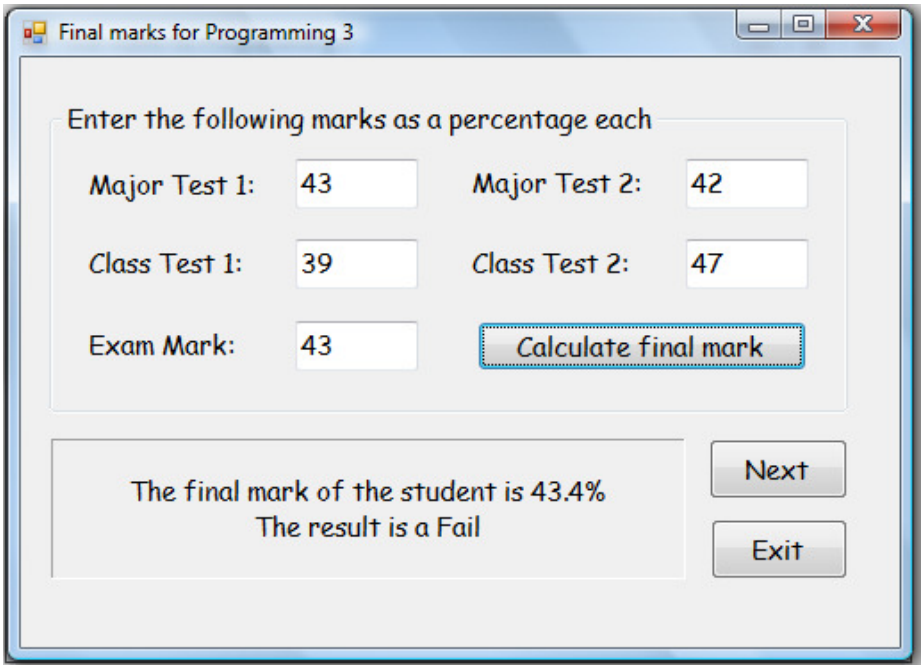
```
Private Sub btnCalcFinalMark_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnCalcFinalMark.Click
   Dim intTest1, intTest2 As Integer
   Dim intClTest1, intClTest2, intHighestClassTest As Integer
   Dim intExam As Integer
   Integer.TryParse(txtTest1.Text, intTest1)
   Integer.TryParse(txtTest2.Text, intTest2)
   Integer.TryParse(txtClTest1.Text, intClTest1)
   Integer.TryParse(txtClTest2.Text, intClTest2)
   Integer.TryParse(txtExam.Text, intExam)
   intHighestClassTest = CalcClMark(intClTest1, intClTest2)
   Call CalcValues(intTest1, intTest2, intHighestClassTest, intExam)
End Sub

Private Sub btnNextStudent_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNextStudent.Click
   txtTest1.Text = ""
   txtTest2.Text = ""
   txtClTest1.Text = ""
   txtClTest2.Text = ""
   txtExam.Text = ""
   lblAnswer.Text = ""
   txtTest1.Focus()
End Sub

Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnExit.Click
   Close()
End Sub
End Class
```

Examples of output:



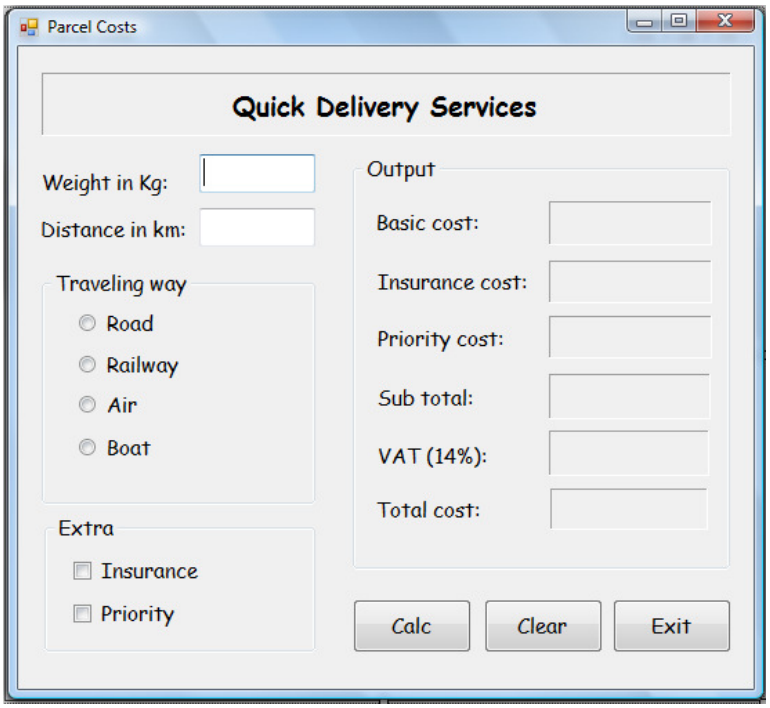


2.5 FUNCTION AND SUB PROCEDURES WITHOUT PARAMETERS

It is possible that a function or a sub procedure can be called without sending any parameters to it.

2.5.1 EXAMPLE 1

Study the following user interface for the courier service called QUICK DELIVERY SERVICES.



The interface is used for an application to calculate parcel costs. The basic cost of a parcel is determined by the weight of the parcel, the way the parcel will be sent as well as the travelling distance. The basic cost is calculated as R1.23 per kg multiplied by the distance and the travelling cost according to the tariffs in the following table:

Road	R0.15
Railway	R0.12
Air	R0.36
Boat	R0.25

If the client chooses that the parcel must be insured, the insurance cost will be 11% of the basic cost. If the client indicates that the parcel is a priority, an additional 15% will be added to the basic cost. 14% VAT must be added to calculate the total cost of the parcel.

When the clear button is clicked, all the controls must be cleared, except the road radio button which will always be the default radio button. However, the user can ignore the clear button and only enter a new weight in kg, in which case all other controls must also be cleared and the road radio button must also be set as the default button.

The code for these two buttons would be as follows:

```
Private Sub btnClear_Click(. . .) Handles btnClear.Click
   txtWeight.Text = ""
   txtDistance.Text = ""
   lblBasicCost.Text = ""
   lblInsuranceCost.Text = ""
   lblPriorityCost.Text = ""
   lblSubTotal.Text = ""
   lblVat.Text = ""
   lblTotalCost.Text = ""
   radRoad.Checked = True
   radRail.Checked = False
   radAir.Checked = False
   radBoat.Checked = False
   chkInsurance.Checked = False
   chkPriority.Checked = False
   txtWeight.Focus()
End Sub

Private Sub txtWeight_TextChanged(. . .) Handles txtWeight.TextChanged
   txtDistance.Text = ""
   lblBasicCost.Text = ""
   lblInsuranceCost.Text = ""
   lblPriorityCost.Text = ""
   lblSubTotal.Text = ""
   lblVat.Text = ""
   lblTotalCost.Text = ""
   radRoad.Checked = True
   radRail.Checked = False
   radAir.Checked = False
   radBoat.Checked = False
   chkInsurance.Checked = False
   chkPriority.Checked = False
   txtWeight.Focus()
End Sub
```

As we can see, there are quite a number of controls that need to be cleared in each case. These two modules are also identical, except for that the Weight text box must also be cleared if the clear button is clicked. In a case like this, the identical code can be placed in a sub procedure and both button click events can simply call it when needed. It can therefore save a lot of code as can be seen in the following example:

```
Private Sub ClearControls()
   txtDistance.Text = ""
   lblBasicCost.Text = ""
   lblInsuranceCost.Text = ""
   lblPriorityCost.Text = ""
   lblSubTotal.Text = ""
   lblVat.Text = ""
   lblTotalCost.Text = ""
   radRoad.Checked = True
   radRail.Checked = False
   radAir.Checked = False
   radBoat.Checked = False
   chkInsurance.Checked = False
   chkPriority.Checked = False
   txtWeight.Focus()
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   txtWeight.Text = ""
   Call ClearControls()
End Sub

Private Sub txtWeight_TextChanged(. . .) Handles txtWeight.TextChanged
   Call ClearControls()
End Sub
```

2.5.2 EXAMPLE 2

If a function or sub procedure is called without sending any parameters to it and it must use variables in order to do calculations, these variables will have to be declared as global variables with module scope, otherwise the function or sub procedure will not be able to get hold of these values.

However, **the use of global variables is not recommended** because unintentional errors can occur. Fewer errors occur if the scope of variables is reduced or when the programmer takes control by sending it only to the function or procedure where it is needed.

In this example we are going to rewrite the program for Reginald who went to a shop to buy a number of fruit bars. Once again, the number bought and the price of a fruit bar is entered in a main procedure. A function is called to calculate the amount due. However, the numbers are declared with module scope and are not sent to the function as parameters. The amount due must include 14% VAT. The answer must be returned to the main procedure where it must be displayed on the screen.

The user interface and output will remain unchanged.

Code in the code editor:

```
Public Class Shopping

' Global variables needed by the function as well as the button click event
   Private intNoItems As Integer
   Private decPrice As Decimal

Private Function CalcAmtDue() As Decimal
   Return Convert.ToDecimal(intNoItems) * decPrice * 1.14D
End Function

Private Sub btnCalc_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCalc.Click
   Dim decAmtDue As Decimal
   Integer.TryParse(txtNoItems.text, intNoItems)
   Decimal.TryParse(txtPrice.text, decPrice)
   decAmtDue = CalcAmtDue()
   lblAnswer.Text = "The amount due = R" & decAmtDue.ToString("N2")
End Sub

Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnClear.Click
   txtNoItems.Text = ""
   txtPrice.Text = ""
   lblAnswer.Text = ""
   txtNoItems.Focus()
End Sub

End Class
```

Take note:
As mentioned earlier, the use of global variables should be minimised and it is recommended to send parameters to a function procedure or sub procedure as needed.



Exercises

Study each of the following sub procedure calls and then write the complete sub procedure in VB.NET as required.

1. Call SortTests(intTest1, intTest2, intTest3)

   The sub procedure must place the best test in the first parameter, the second best test in the second parameter and the lowest mark in the third parameter

2. Call CalcNetSal(decGrossSal, 23.5, decTaxAmt, decNetSal)

   The sub procedure will receive a gross salary and a tax percentage. It must then calculate the tax amount for this employee and the net salary by deducting the tax amount from the gross salary. All values will be applicable to a month's salary. The tax amount and net salary must be available to the calling module after it has been calculated.



Practical

Solve the following problems by first doing the planning and then writing the VB.NET program. Use functions and sub procedures wisely.

1. Ronnie sells newspapers on the corner of Long and Main streets from Mondays to Saturdays. Enter the profit he earns per newspaper at the beginning of the program and then go into a loop to input the number of newspapers he sold per day in a sub procedure. Calculate and display his profit of the week. Use other functions and sub procedures.

2. A fleet of taxis transport passengers from Main Square to different parts of the town. The owner of the taxis needs to know how many passengers travel to region A, region B or region C. He earns 50 cents profit on a passenger travelling to region A, 65 cents to region B and 72 cents to region C. Enter the region for every passenger and the region code 'Z' will terminate the input.
   Calculate how many passengers travel to each of the regions and also calculate the profit earned by the owner. Display the results on the screen. Use functions and/or sub procedures.

3. Max Contractors has 5 employees who each receive a weekly wage. The basic wage is entered. An employee may also borrow money from Max Contractors and 25% of the loan is subtracted from the weekly wage. If this amount to be subtracted is less than R50, then R50 is subtracted. However, if R50 is more than the remainder of the loan amount, the remainder is subtracted. The loan amount is also entered. The program must display the amount the employee will receive as well as the remaining amount of the loan.

CHAPTER 3
ARRAYS

3.1 INTRODUCTION

Quite often, the programmer has to use a number of related variables that are of the same data type and are all used for a specific purpose in a program e.g. the names of the 12 months. These variables may be grouped together in an array with a single name.

OUTCOMES

When you have studied this chapter you should be able to:

- understand what an array is
- declare and initialize a one-dimensional array
- store data in a one-dimensional array
- manipulate a one-dimensional array
  - display elements in the array
  - find the highest and lowest entry in an array
  - search an array to find a specific element
  - access an element in the array
  - compute the total and average of all the elements in an array etc.
- use the For Each.. Next statement to retrieve array elements
- understand and use parallel arrays
- declare, understand and manipulate two-dimensional arrays
- sort an array using the bubble sort method
- sort an array using the selection sort method

3.2 A ONE-DIMENSIONAL ARRAY

An array consists of a number of related variables that has the following properties:

1.2.1 It has a **single name**.
1.2.2 A single variable in the array is called an **element**.
1.2.3 All the elements in an array are of the **same data type**.
1.2.4 The position of an element in the array is called an **index** or a **subscript**.

An array can be seen as a row or a column of elements.

Name of the array

The name of the array must be descriptive

e.g. monthArray that contains the names of the 12 months in a year,
price that contains the prices of the 100 items in a shop,
quantityInStock that contains the quantity in stock of 120 items.

Element

One element in an array is a single variable that contains a single value at a given time:

e.g. One name of a month in monthArray e.g. September,
R25.57 that is the price of an item in the array called price
25 that represents the quantity in stock of a specific item in the array called QuantityInStock.

An element in an array need not have an initial value.

Data type

All the elements of an array must be of the same data type e.g. String, Integer, etc.

e.g. monthArray will be of the type String
price may be of the type Decimal
quantityInStock will be of the type Integer.

Index

The **index** indicates the position of the element in the array. The index is sometimes referred to as a **subscript**. The index must always be 0 or a positive integer.
The index of the 1st element of an array is always 0 (zero) and the index of the last element of the array is (number of elements – 1) e.g. the indexes of the array mathScores that has 30 elements will start at 0 and the last index will be 29.

The monthArray may be visualized as follows:

monthArray

January	February	March	.....	December
Index = 0	1	2		11

or

monthArray(0)	January
monthArray(1)	February
monthArray(2)	March
monthArray(3)	April
monthArray(4)	May
monthArray(5)	June
montArray(6)	July
montArray(7)	August
monthArray(8)	September
monthArray(9)	October
monthArray(10)	November
monthArray(11)	December

To refer to a specific element in the array, the name of the array as well the index of the element is used:
monthArray(0) refers to January and
monthArray (11) refers to December

3.3 DECLARING A ONE-DIMENSIONAL ARRAY IN VB.NET

In VB.NET, you can either declare an array and assign initial values to the elements, or you can simply declare the array structure. In such a case, VB.Net will initialize it (0's for numeric values and an empty string for string values)

Declaring a one dimensional array:

Syntax 1:

```
{Dim | Private} arrayName() As Datatype = {Initial values}
```

Syntax 2:

```
{Dim | Private} arrayName(highestSubscript) As Datatype
```

Example 1:

```
Dim intSalesPersonNumber() As Integer = {20, 13, 25, 66, 52}
```

An array with 5 elements will be declared. The 5 elements will all have the following initial values:

intSalesPersonNumber(0) = 20
intSalesPersonNumber(1) = 13
intSalesPersonNumber(2) = 25
intSalesPersonNumber(3) = 66
intSalesPersonNumber(4) = 52

Example 2:

```
Dim intSalesPersonNumber(4) As Integer
```

An array with 5 elements will be declared. The 5 elements will all have an initial value of 0.

In this case it will be possible to give the same values to the elements by using 5 assignment statements:

Syntax of assignment statement to store data in a one-dimensional array:

```
arrayName(IndexValue) = value
```

Example 2:

```
Dim intSalesPersonNumber(4) As Integer

intSalesPersonNumber(0) = 20
intSalesPersonNumber(1) = 13
intSalesPersonNumber(2) = 25
intSalesPersonNumber(3) = 66
intSalesPersonNumber(4) = 52
```

Although values can be assigned to array elements in this way, it is normally more efficient to do it in a loop, as indicated in the next section.

3.4 ADVANTAGES OF USING A ONE DIMENSIONAL ARRAY

Arrays provide a way to improve code and to make it shorter and more efficient. Let us assume that a student has written 10 tests in one subject. The final mark must be calculated as the average of these 10 test marks.

If we only had simple variables (also called scalar variables), we must use 10 different variables in which the 10 test marks must be stored. Values for these 10 variables must then be entered 10 times in order to use it to calculate the final mark for the student.

The algorithm would look as follows:

```
CalculateFinalMark
        Display "Please provide the percentage for test 1"
        Enter T1
        Display "Please provide the percentage for test 2"
        Enter T2
        Display "Please provide the percentage for test 3"
        Enter T3
        Display "Please provide the percentage for test 4"
        Enter T4
        Display "Please provide the percentage for test 5"
        Enter T5
        Display "Please provide the percentage for test 6"
        Enter T6
        Display "Please provide the percentage for test 7"
        Enter T7
        Display "Please provide the percentage for test 8"
        Enter T8
        Display "Please provide the percentage for test 9"
        Enter T9
        Display "Please provide the percentage for test 10"
        Enter T10
        finalMark = (T1 + T2 + T3 + T4 + T5 + T6 + T7 + T8 + T9 + T10) / 10
        Display "The final mark for the student = ", finalMark, " %"
End
```

It is clear that this method is not effective at all. The code would even be worse if the student wrote 15 or 20 tests. It is possible, with our knowledge of loops at this stage, that we could rewrite the code as follows:

```
CalculateFinalMark
        totalMark = 0
        for T = 1 to 10
                display "Please provide the percentage for test ", T
                enter testMark
                totalMark = totalMark + testMark
        next
        finalMark = totalMark  / 10
        display "The final mark for the student = ", finalMark, " %"
End
```

Yes, it surely is more efficient to enter the next test mark during every execution of the loop and to accumulate the total for all the tests within the loop. However, the individual marks will not be available at a later stage for any other references because every time a new mark is entered, the previous value of the variable (the previous mark) is replaced by the new mark.

Because the marks are all of the same data type and the marks are related to one another (all the marks are for the same student and will contribute to the calculation of the final mark) it can also be stored in an array.

The code to store data in an array is similar to the previous code. However, instead of entering data into one variable, it is entered into an element of the array. In this example, the index of the first element is 0 and that of the last element is 9 because there are 10 tests to be entered. The array contains 10 elements that each represents a different test mark.

```
For T = 0 to 9
        display "Please provide the percentage for test ", T + 1
        enter testMark(T)
        totalMark = totalMark + testMark(T)
Endfor
```

After the data has been stored in the elements of an array, it can be manipulated in a way similar to manipulation of normal scalar variables. All the elements in the array structure or only specific elements in the structure can be used in calculations. The contents of an array element may also be changed or displayed etc.

It is possible to find the element with the highest or lowest value in the array, to sort the elements in a specific order, to calculate the average of all the elements and much more.

A few examples of manipulation of elements in an array will be demonstrated.

3.5 EXAMPLES OF A ONE-DIMENSIONAL ARRAY

3.5.1 EXAMPLE 1

Problem:

Store the 10 test marks for the student in an array and then calculate the average of the 10 test marks in order to determine the final mark of the student. After the final mark has been calculated, all the marks as well as the final mark must be displayed.

Planning:

	Description	Type	Name of variable / array
Input:	10 Test marks entered in an array	integer	testMark
Output:	10 Test marks entered in an array	integer	testMark
	Final Mark	integer	finalMark

Algorithm:

```
CalculateFinalMark
        totalMark = 0
        for T = 0 to 9
                display "Please provide the percentage for test ", T + 1
                enter testMark(T)
                totalMark = totalMark + testMark(T)
        next
        finalMark = totalMark  / 10
        for T = 0 to 9
                display "Test ", T+ 1, " = ", testMark(T)
        next
        display "The final mark for the student = ", finalMark, "%"
End
```

After the algorithm has been executed, the variables can be visualized as follows:

testMark(0)	60
testMark(1)	65
testMark(2)	77
testMark(3)	45
testMark(4)	87
testMark(5)	93
testMark(6)	74
testMark(7)	39
testMark(8)	45
testMark(9)	70

totalMark	655
finalMark	65.5

VB.NET Application:

Code in the code editor:

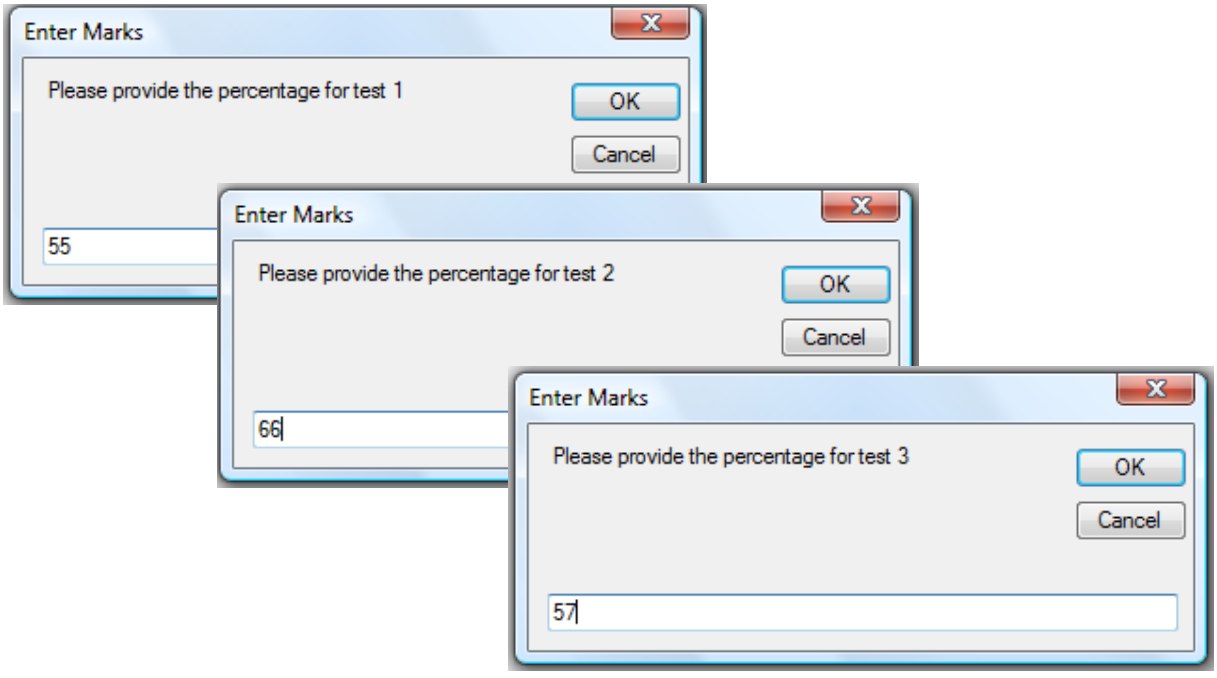
```
Public Class TestMarks

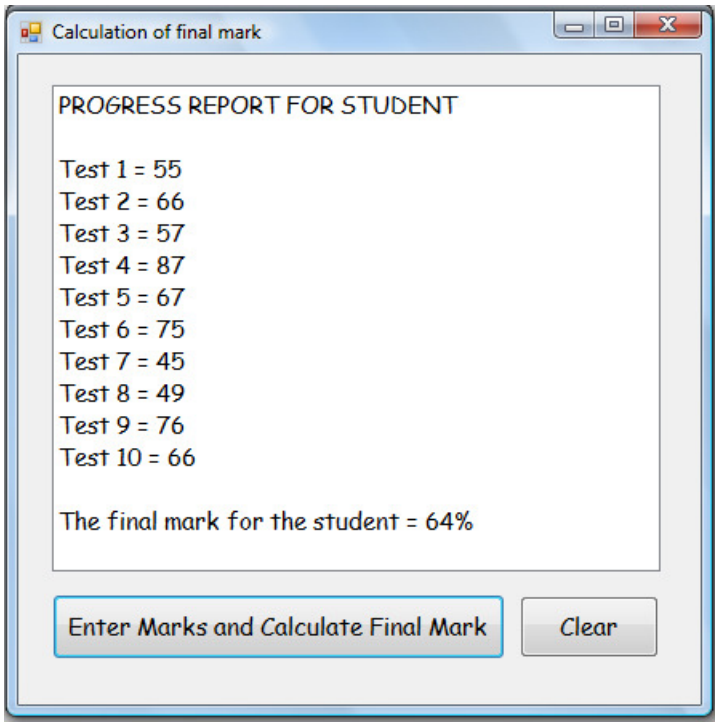
Private Sub btnEnter_Click(. . .) Handles btnEnter.Click
   Dim intTotalMark As Integer = 0
   Dim strTestMark As String
   Dim intTestMark(9) As Integer
   Dim decFinalMark As Decimal
   Dim T As Integer
   For T = 0 To 9
       strTestMark = InputBox("Please provide the percentage for test " & _
                              T + 1, "Enter Marks")
       Integer.TryParse(strTestMark, intTestMark(T))
       intTotalMark = intTotalMark + intTestMark(T)
   Next
   decFinalMark = Convert.ToDecimal(intTotalMark / 10)
   lstTestMarks.Items.Add("PROGRESS REPORT FOR STUDENT")
   lstTestMarks.Items.Add(" ")
   For T = 0 To 9
       lstTestMarks.Items.Add("Test " & T + 1 & " = " & intTestMark(T))
   Next
   lstTestMarks.Items.Add(" ")
   lstTestMarks.Items.Add("The final mark for the student = " & _
                          decFinalMark.ToString("F0") & "%")
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   lstTestMarks.Items.Clear()
End Sub
End Class
```

Example of output:

Enter all 10 test marks via input boxes, e.g.





3.5.2 EXAMPLE 2:

Problem:

Rewrite the previous example, but calculate the final mark for a student as the average of the 9 best marks. In this case, the lowest mark must be determined and subtracted from the total before the average is calculated.

Algorithm:

```
CalculateFinalMark
        totalMark = 0
        for T = 0 to 9
                display "Please provide the percentage for test ", T+ 1
                enter testMark(T)
                totalMark = totalMark +testMark(T)
        next
        lowestMark = testMark(0)
        for T = 1 to 9
                if testMark(T) < lowestMark
                        lowestMark = testMark(T)
                endif
        next
        totalMark = totalMark – lowestMark
        finalMark = totalMark  / 9
        for T = 0 to 9
                display "Test ", T+ 1, " = ", testMark(T)
        next
        display "The final mark for the student = ", finalMark, "%"
End
```

All 10 marks will still be stored in the array and will be available for later reference. The total of all the marks is once again accumulated within the loop. However, the lowest test mark is determined in a second loop. After this loop has been executed, this lowest test mark is deducted from the total to exclude this mark. The total is then divided by 9 in order to determine the average or final mark. The lowest test mark could also have been determined in the first loop.

This VB.NET program includes validation. Every test mark entered is validated to ensure that it contains a valid integer from 0 to 100. If not, an appropriate error message is displayed and the test mark must be re-entered.

VB.NET Application:

Code in the code editor:

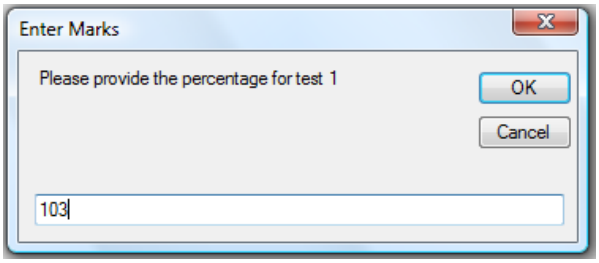
```
Public Class TestMarks

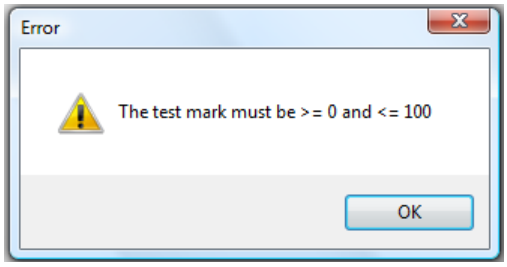
Private Sub btnEnter_Click(. . .) Handles btnEnter.Click
   Dim intTotalMark As Integer = 0
   Dim strTestMark As String
   Dim intTestMark(9) As Integer
   Dim intLowestMark As Integer
   Dim decFinalMark As Decimal
   Dim blnValidTestMark As Boolean
   Dim T As Integer
   For T = 0 To 9
      Do
         strTestMark = InputBox("Please provide the percentage for test " _
                                & T + 1, "Enter Marks")
         blnValidTestMark = Integer.TryParse(strTestMark, intTestMark(T))
         If blnValidTestMark = True Then
            If intTestMark(T) >= 0 AndAlso intTestMark(T) <= 100 Then
               intTotalMark = intTotalMark + intTestMark(T)
            Else
               MessageBox.Show("The test mark must be >= 0 and <= 100", _
                 "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
               blnValidTestMark = False
            End If
         Else
            MessageBox.Show("The test mark must be an integer", "Error", _
                            MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
            blnValidTestMark = False
         End If
      Loop Until blnValidTestMark
   Next
   intLowestMark = intTestMark(0)
   For T = 0 To 9
       If intTestMark(T) < intLowestMark Then
          intLowestMark = intTestMark(T)
       End If
   Next
   intTotalMark = intTotalMark - intLowestMark
   decFinalMark = Convert.ToDecimal(intTotalMark / 9)
   lstTestMarks.Items.Add("PROGRESS REPORT FOR STUDENT")
   lstTestMarks.Items.Add(" ")
   For T = 0 To 9
       lstTestMarks.Items.Add("Test " & T + 1 & " = " & intTestMark(T))
   Next
```

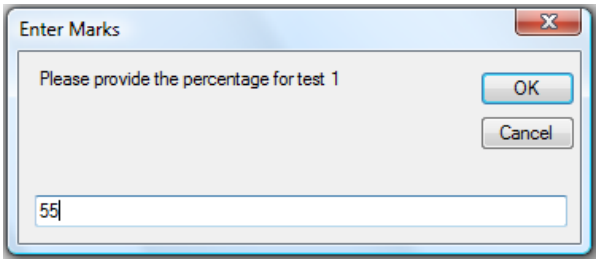
```
   lstTestMarks.Items.Add(" ")
   lstTestMarks.Items.Add("The final mark for the student = " & _
                          decFinalMark.ToString("F0") & "%")
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   lstTestMarks.Items.Clear()
End Sub
End Class
```

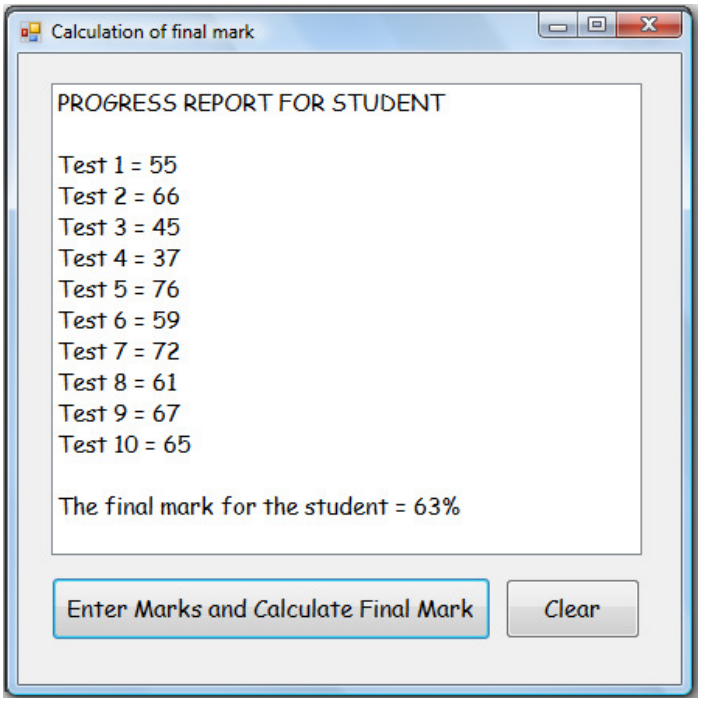
Example of output:







etc. . . .



Take note:
In this example, the 37% will be ignored and not taken into account when the final mark is calculated.

3.5.3 EXAMPLE 3

Problem:

The identity numbers of all the employees of a company is stored in an array. The manager wants to know if a woman whose identity number he has to enter is an employee of the company. Display the result on the screen. Plan and write a VB.NET program to solve this problem. For the sake of testing the program, use an array of only 8 identity numbers.

Planning:

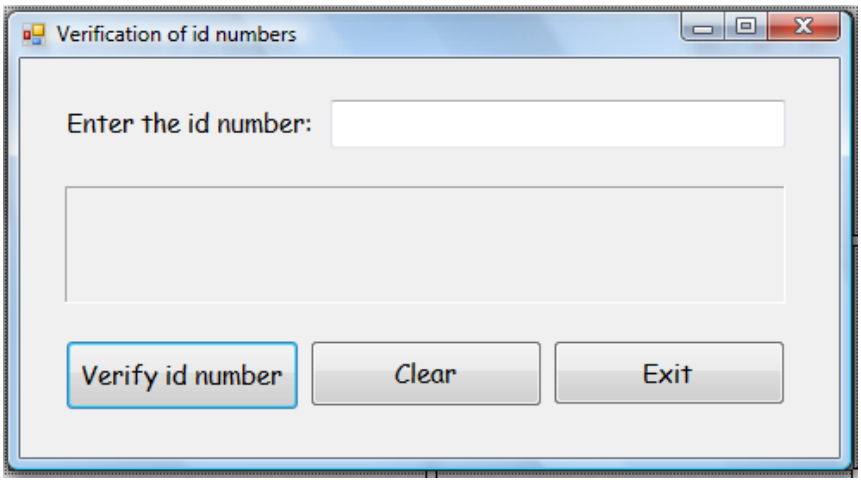
	Description	Type	Name of variable
Array containing the identity numbers: empIds			
Input:	identity number	integer	idNo
Output:	result of search	string	- none -

Algorithm:

```
FindEmployee
        boolFound = False
        display "Enter the identity number of the employee"
        enter idNo
        for m = 0 to 7
                if empIds(m) = idNo                ~ empIds is an array with 8 elements
                        boolFound = True           ~ The person is found!
                endif
        next
        if boolFound = True
                display  "The person whose id number is ", idNo, " is an employee"
        else
                display "The person whose id number is ", idNo, " is not an employee"
        endif
end
```

VB.NET Application:

User Interface:



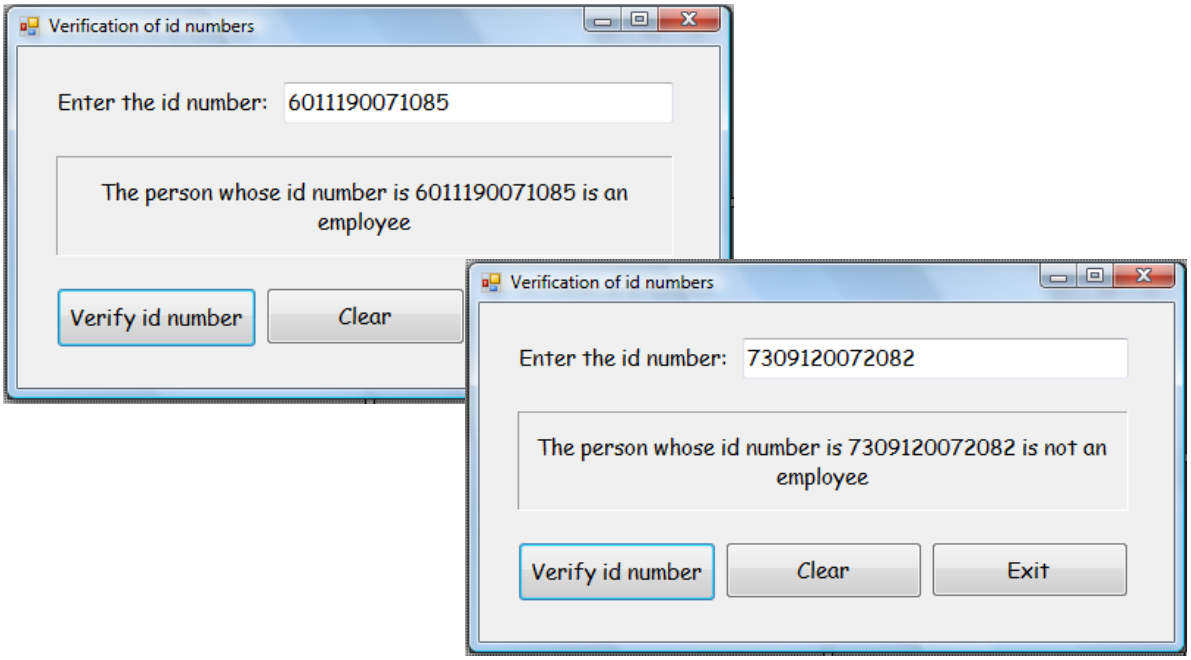
Code in the code editor:

```
Public Class FindEmployee

Private Sub btnVerify_Click(. . .) Handles btnVerify.Click
   Dim strEmpIds() As String = {7503040073085, 6011190071085, _
            6712140092082, 8005060083085, 907080077083, 6904090077082, _
            8210180074084, 8809080072081}
   Dim blnFound As Boolean = False
   Dim strIdNo As String
   strIdNo = txtIdNum.Text
   For m As Integer = 0 To 7
       If strEmpIds(m) = strIdNo Then
          blnFound = True
          Exit For
       End If
   Next
   If blnFound = True Then
      lblAnswer.Text = "The person whose id number is " & strIdNo & _
                       " is an employee"
   Else
      lblAnswer.Text = "The person whose id number is " & strIdNo & _
                       " is not an employee"
   End If
End Sub
Private Sub btnClear_Click(. . .) Handles btnClear.Click
   txtIdNum.Text = ""
   lblAnswer.Text = ""
   txtIdNum.Focus()
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Output Examples:



3.5.4 EXAMPLE 4

Problem:

Fill an array called *evenNumbers* that has 44 elements with consecutive even numbers starting at 14.

Algorithm:

FillArray

~This program fills the array with consecutive even numbers starting at 14.
declare an array evenNumbers with 44 elements of the type integer
number = 14
for i = 0 to 43
evenNumbers(i) = number
number = number + 2
next i
end

VB.NET Application:

The array elements will not only be loaded in the array, but also displayed on a label.

Code in the code editor:

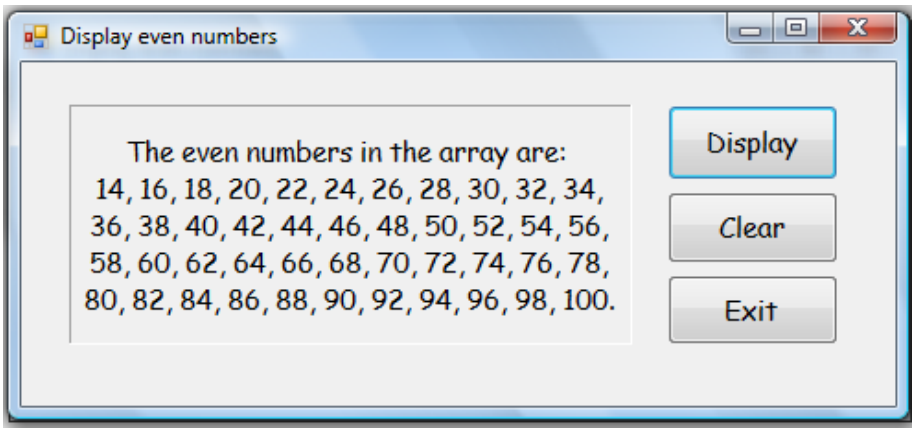
```
Public Class FillArray

Private Sub btnDisplay_Click(. . .) Handles btnDisplay.Click
   Dim intEvenNumbers(43) As Integer
   Dim intNumber As Integer = 14
   For i As Integer = 0 To 43
       intEvenNumbers(i) = intNumber
       intNumber += 2
   Next
   lblAnswer.Text = "The even numbers in the array are: " & _
                    ControlChars.NewLine
   For x As Integer = 0 To 43
       lblAnswer.Text = lblAnswer.Text & intEvenNumbers(x)
       If x < 43 Then
          lblAnswer.Text = lblAnswer.Text & ", "
       Else
          lblAnswer.Text = lblAnswer.Text & "."
       End If
   Next
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   lblAnswer.Text = ""
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Output:



Modify the example to start at any even number.

Algorithm:

```
FillArray
        display "Enter the first even number with which to fill the array"
        enter firstNumber
        if firstNumber is numeric
                if firstNumber Mod 2 = 0              ~ Test if it is an even number
                        number = firstNumber
                        for i = 0 to 43
                                evenNumbers(i) = number
                                number = number + 2
                        next i
                else
                        display "The number must be an even number"
                endif
        else
                display "The number must be an integer"
        endif
end
```

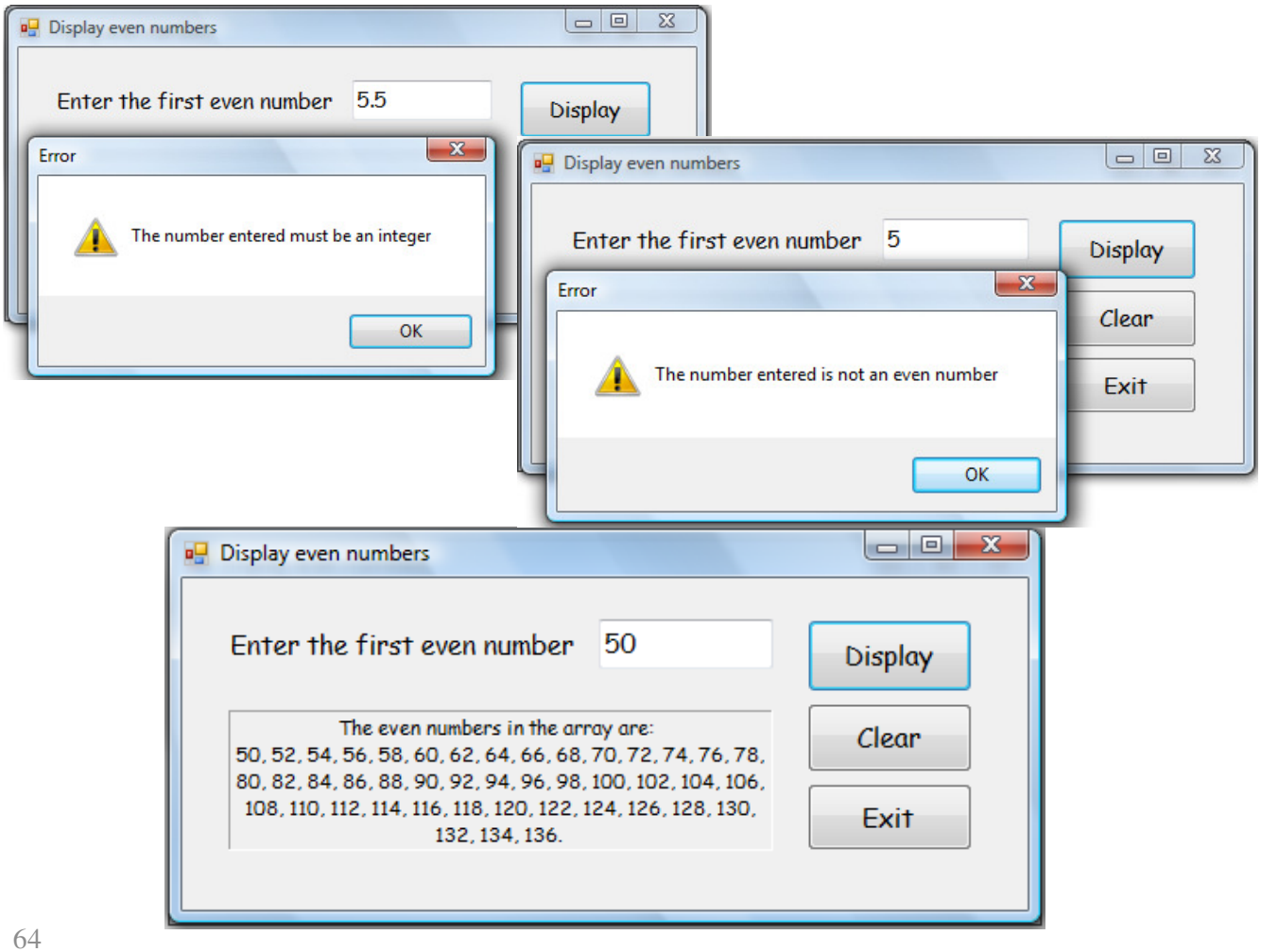
Once again, the VB.NET program will not only assign the values to the array elements, but also display it on a label.

Code in the code editor for the display button:

```
Private Sub btnDisplay_Click(. . .) Handles btnDisplay.Click
   Dim intEvenNumbers(43) As Integer
   Dim intFirstEvenNumber, intNumber As Integer
   Dim blnValidInteger As Boolean
   blnValidInteger = Integer.TryParse(txtEvenNum.Text, intFirstEvenNumber)
   If blnValidInteger Then
      If intFirstEvenNumber Mod 2 = 0 Then
         intNumber = intFirstEvenNumber
```

```
            For i As Integer = 0 To 43
                intEvenNumbers(i) = intNumber
                intNumber += 2
            Next
            lblAnswer.Text = "The even numbers in the array are: " & _
                             ControlChars.NewLine
            For x As Integer = 0 To 43
                lblAnswer.Text = lblAnswer.Text & intEvenNumbers(x)
                If x < 43 Then
                   lblAnswer.Text = lblAnswer.Text & ", "
                Else
                   lblAnswer.Text = lblAnswer.Text & "."
                End If
            Next
         Else
            MessageBox.Show("The number entered is not an even number", _
                  "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
            txtEvenNum.Text = ""
            txtEvenNum.Focus()
         End If
      Else
         MessageBox.Show("The number entered must be an integer", "Error", _
                         MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
         txtEvenNum.Text = ""
         txtEvenNum.Focus()
      End If
   End Sub
```

Examples of output:



3.5.5 EXAMPLE 5

Problem:

An array called *noInStock* already contains valid positive integer values in the 20 elements. Declare the array using the following values in the elements: 4, 15, 38, 53, 8, 95, 129, 653, 45, 658, 12, 40, 0, 98, 74, 32, 9, 39, 25, 87. Find and display the index of the highest number as well as the highest number in *noInStock.*

Algorithm:

FindHighest

Declare the array noInStock as integer using the values
4, 15, 38, 53, 8, 95, 129, 653, 45, 658, 12, 45, 0, 98, 74, 32, 9, 39, 25, 87.

~ Compare all the values in the array to find and display the highest value

highest = noInStock(0) ~ assign first value to highest
for k = 1 to 19
if noInStock(k) > highest
highest = noInStock(k)
foundAtIndex = k
endif
next k

~ After testing the values in all the elements, the highest can now be displayed.

display "The highest value in the array noInStock is: ", highest
display "It was found at index poision " fountAtIndex
end

Please note that the index of first value used in the loop structure is not that of the first element, because the value of the first element has already been assigned to the variable called highest.

Code in the code editor:

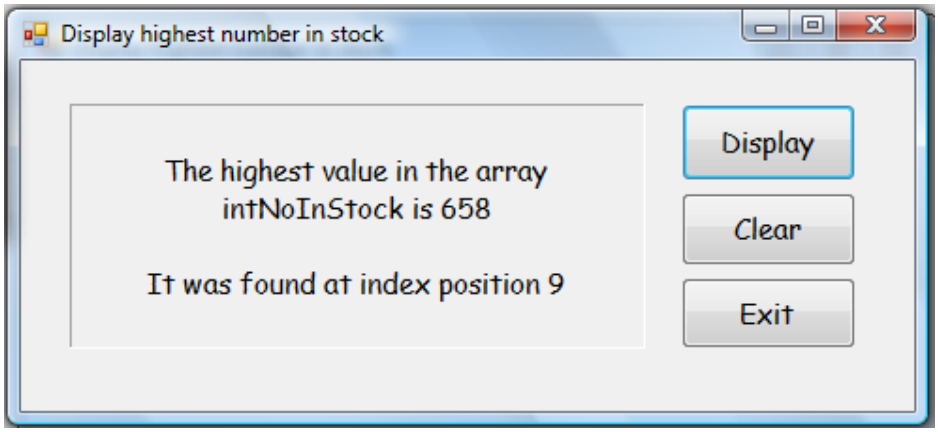
```
Public Class noInStock

Private Sub btnDisplay_Click(. . .) Handles btnDisplay.Click
   Dim intNoInStock() As Integer = {4, 15, 38, 53, 8, 95, 129, 653, 45, _
                                    658, 12, 40, 0, 98, 74, 32, 9, 39, 25, 87}
   Dim intHighest As Integer = intNoInStock(0)
   Dim intFoundAtIndex As Integer
   For k As Integer = 1 To 19
       If intNoInStock(k) > intHighest Then
          intHighest = intNoInStock(k)
          intFoundAtIndex = k
       End If
   Next
   lblAnswer.Text = "The highest value in the array intNoInStock is " _
            & intHighest.ToString & ControlChars.NewLine & _
            ControlChars.NewLine & "It was found at index position " _
            & intFoundAtIndex.ToString
End Sub
```

```
Private Sub btnClear_Click(. . .) Handles btnClear.Click
   lblAnswer.Text = ""
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Output:





Exercises

1. The shop called Best Buys stores all the prices of the 30 different items sold in the shop in an array. Declare the array called *priceArray* in VB.NET. The array must be initialized when it is declared.

2. Declare an array in VB.NET with the following 7 product codes: AB101, CD202, DF321, GR345, HT191, YR293, and KL445.

3. Write an algorithm for the following:
   The array called *numbers* contains 35 valid integer numbers. Determine and display how many of these values are greater than the average value of all the values of the elements. Hint: Calculate the average before counting the number of higher values.

4. Declare an array called *abcArray* with 20 integer elements. Populate the array using the values of the series 1, 2, 4, 7, 11, ….. Write code to replace the value of every fourth element by the sum of the previous 3 elements. Display the values of all the new elements.

5. Declare an array with 20 elements. This array may contain positive integers. Accept a value for every element from the keyboard. Validate the value before storing it in the array element. Display all elements entered, the highest value entered, the lowest value entered as well as the average of all the numbers entered.



1. An array called letterArray has 30 elements that contains a sentence with one letter per element e.g. TODAY IS A LOVELY SUNNY DAY

T	O	D	A	Y		I	S		A		L	O	V	E	L	Y		S	etc.

You are asked to write a VB program to count how many elements contain either the letter A or E. Display the count as well as what the percentage A and E letter is of the total number of letters.

The algorithm to solve this problem is provided. Write the corresponding VB.Net program.

CountAE

```
        count = 0
        for k = 0 to 29
                if letterArray(k) = "A"  OR letterArray(k) = "E"
                        count = count + 1
                endif
        next

        percentage = count / 30 * 100
        display "There are " , count, "A's and E's in the array"
        display "The A's and E's are ", percentage, "% of the total number of letters"
end
```

2. The shopkeeper Joe Smart bought 50 items of every of the 100 items he wants to sell in his new shop called Joe's Specials. Declare an array for the number in stock of every product numbered from 1 to 100 and initialise the value of every element to 50. An unknown number of customers buy products during the day and the product number (1 – 100) is entered as well as the number of products bought. When the shop closes, a product number of 999 is entered to indicate the end of the transactions.

   Every time a transaction takes place, the number the customer buys, is compared to the number on hand, and if enough items are in stock, the transaction proceeds. If the stock is not enough, do not proceed with the transaction, but display a suitable message. The number bought is subtracted from the number on hand. If the remaining number is less than 10, a message must be displayed that shows the number of the product and the new quantity on hand.

   At the end of the day, the total number of products sold during the day is displayed. Plan the solution for this problem and implement it in VB.NET.

3. The array *account* has 6 elements that contains outstanding amounts of money representing amounts owed to the shop Fancy Goods by a single client. The first element contains the amount bought during the current month. The next element contains the amount outstanding for last month, etc. The sixth element contains an amount in arrears for 6 months plus any other amounts in arrear (e.g. month 7, 8, etc)

   Examples to illustrate possible arrays:

   account

42.75	0.00	12.50	150.34	81.20	433.12

   This customer bought goods to the value of R42.75 during the current month, nothing during the previous month, other amounts for the next months, but still owes R433.12 for the sixth and/or more months.

   The next example is the account of a client who pays the full amount every month:

   account

124.20	0.00	0.00	0.00	0.00	0.00

   Plan an algorithm to solve the following problem and then implement the corresponding VB.NET application.

   Declare the array using the first set of values. Ask the client how much he/she is paying towards the account. This amount must be subtracted from the longest amount in arrears. If the payment is more than the longest amount in arrears, the remainder is subtracted from the next amount in arrears, etc. If the client pays more than all total amounts due, a negative amount must be shown in the first element. Display the new values of every element on the screen.

4. Use the same array as in the question above (question 3). Prepare the array for the new month by adding the values of elements 5 and 6 to store it in element 6. Move the amounts in all the other elements one position up i.e. 4 to 5, 3 to 4, etc. Initialise the first element to prepare for the new month. But, if the previous first element contained a negative amount, this amount must be moved to the 1st element, while the 2nd element is initialised.

3.6 For Each . . Next Statement in VB.NET

This statement can be used if every element in the array must be updated or retrieved. Instead of using a normal for .. next statement and a subscript or index, every *element* in the *group* is visited. Element is a variable that is declared to keep track of every array variable and group is the name of the array.

Syntax :

```
For Each element [As Datatype] In group
     [statements]
Next element
```

Example 1:

```
Dim strMonthName() As String = {"January", "February", "March", _
          "April", "May", "June", "July", "August", "September", _
          "October", "November", "December"}
For each strMonth As String in strMonthName
     lstDisplay.Items.Add(strMonth)
Next
```

3.7 Example with For Each Next Statement:

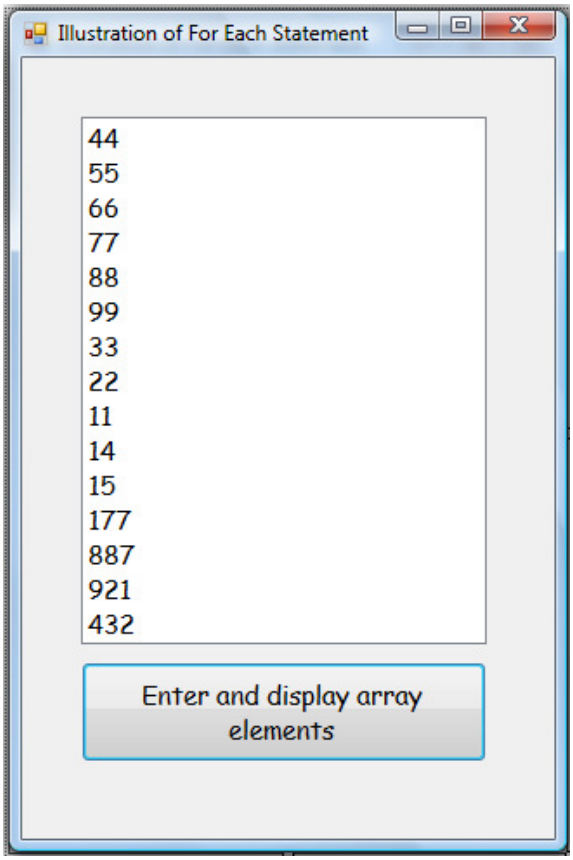
Write a VB.NET program to enter 15 integer values and to store it in an array. Display the values in the array on a list box by making use of the For Each . . Next statement.

Code in the code editor:

```
Public Class ForEach

Private Sub btnDisplay_Click(. . .) Handles btnDisplay.Click
   Dim intNumbers(14) As Integer
   For x As Integer = 0 To 14
       intNumbers(x) = Convert.ToInt32(InputBox("Enter number " & x + 1 ))
   Next
   For Each intNum As Integer In intNumbers
       lstDisplay.Items.Add(intNum)
   Next
End Sub
End Class
```

Output:





Questions

Can the For Each . . Next statement be used to enter store values in an array?

3.8 Parallel (paired) arrays

Parallel arrays are 2 or more arrays relating to one another and have the same number of elements. The elements of these arrays need not be of the same type. It will be explained by using an example where 2 arrays are called *code* and *description.*

The array called *code* will contain the code of an item, while the array called *description* will contain the description of an item in the corresponding element i.e. the indexes of the 2 arrays will be equivalent for a specific item.

In the following table the codes with the corresponding description are given:

code	**description**
ab3	chair
kh8	table
vs1	bed
gp5	cupboard
wd4	sofa

The 2 arrays can be represented as follows:

code

ab3	kh8	vs1	gp5	wd4
Index: 0	1	2	3	4

description

chair	table	bed	cupboard	sofa
Index: 0	1	2	3	4

If we know what the value of a code of an element is, we need to find the index of this value and then go to the *description* array to find the description of the item in the element with the same index.

3.9 Example with parallel arrays

Problem:

Declare the 2 arrays as shown above. Write an algorithm to enter a code, find the corresponding description and then display the description on the screen. If the description is not found (i.e. the array *code* does not contain the entered code) a suitable message must be displayed.

Algorithm:

FindDescription

~ Find the description for an entered item code.

Declare the arrays *code* and *description* using the given values
itemIndex = 0

display "Provide the code of an item"
enter itemCode

Do While itemIndex < 5 AND code(itemIndex) <> itemCode
ItemIndex = itemIndex + 1
Loop

~ Display the answer

if itemIndex < 5
display "The description for ", itemCode, " is ", description(itemIndex)
else
display "You entered an incorrect item code"
endif
end

Code in the code editor:

```
Public Class FindDescription

Private Sub btnDisplay_Click(. . .) Handles btnDisplay.Click

   Dim intItemIndex = 0

   Dim strCode() As String = {"ab3", "kh8", "vs1", "gp5", "wd4"}
   Dim strDescription() As String = {"chair", "table", "bed", _
                                     "cupboard", "sofa"}

   Dim strItemCode As String
   strItemCode = txtCode.Text

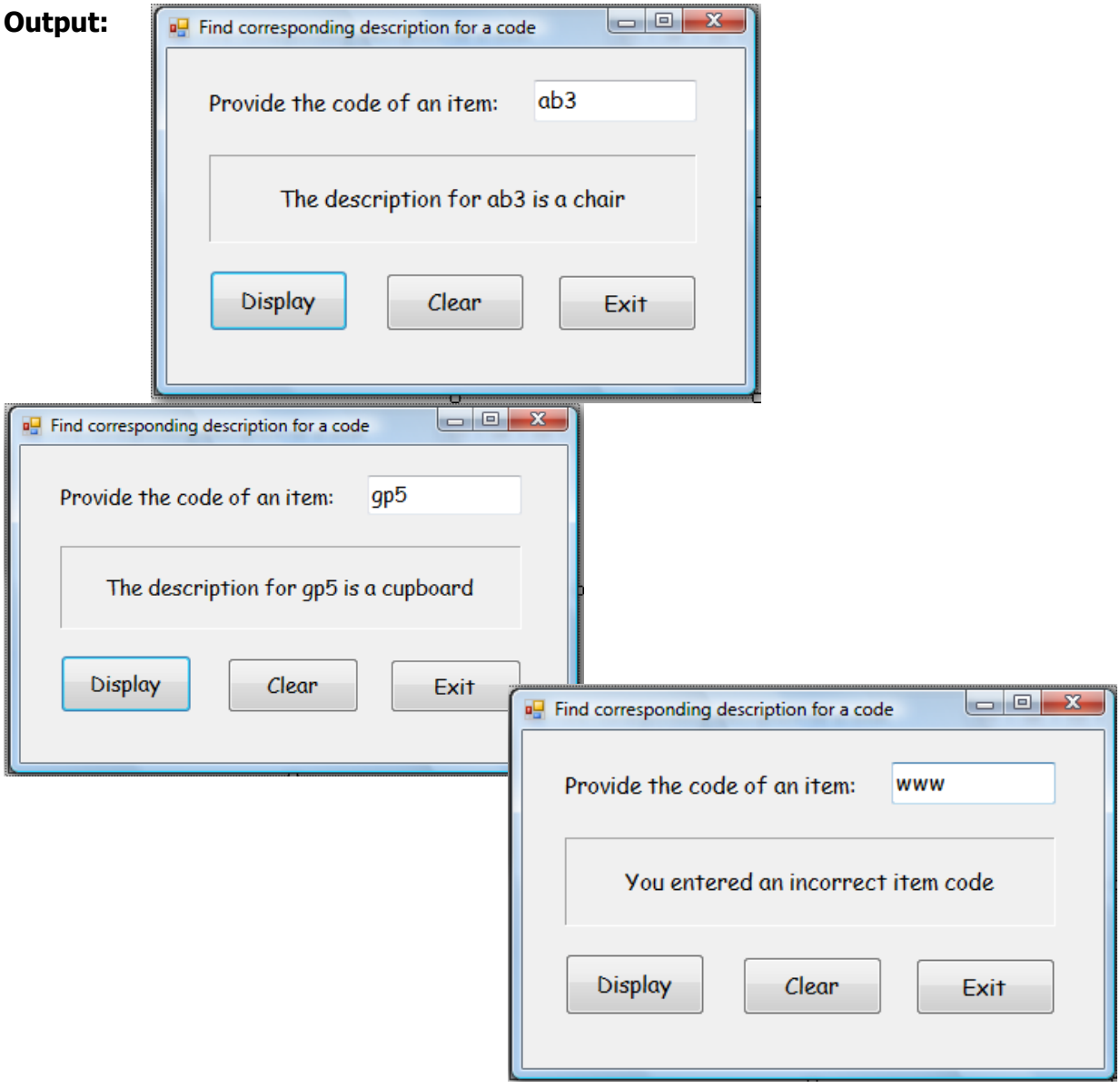
   Do While intItemIndex < 5 AndAlso strCode(intItemIndex) <> strItemCode
      intItemIndex += 1
   Loop
```

```
   If intItemIndex < 5 Then
      lblAnswer.Text = "The description for " & strItemCode & " is a " _
                       & strDescription(intItemIndex)
   Else
      lblAnswer.Text = "You entered an incorrect item code"
      txtCode.Text = ""
      txtCode.Focus()
   End If
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   lblAnswer.Text = ""
   txtCode.Text = ""
   txtCode.Focus()
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Output:





Solve the following problems by doing the planning and then writing the VB.NET program:

1. Using the values in the following table, declare 3 parallel arrays.

Code of snack	Description	Price in R
75	Chocolate	5.75
23	Soda	6.25
89	Pizza slice	8.50
41	Burger	15.85
17	Muffin	7.35

The user is asked to enter the code of the snack he/she wants to buy as well as the number of these snacks. The program must find the description and the price in the arrays, calculate the amount, add 14% vat and then display the number bought, description and total amount due on the screen.

2. Six children took part in a series of 3 quizzes each. Declare an array to contain the name of every child and another 3 parallel arrays that can contain the points that every child scored in each quiz. The values are entered by the user. Calculate the average number of points obtained by every child and store these values in a 5th array. Determine and display the name of the child that obtained the most average points.

3.10 Two-dimensional arrays

A one-dimensional array consists of one row that contains a number of columns e.g. the minimum temperature for the past 7 days of the week as can be seen in the following diagram:

3	4	5	3	2	3	6

This one-dimensional array can be expanded to a two-dimensional array when the maximum temperature for every day is added as follows:

3	18
4	22
5	23
3	21
2	24
3	19
6	25

The particulars of a day can be found in a row i.e. in row 0 the minimum temperature and maximum temperature for day 1. Similarly, row 3 column 0 contains the minimum temperature for day 4 and row 3 column 1 the maximum temperature for day 4.

We can now name the array *temperatures* that consists of 7 rows and 2 columns. When addressing an element the row number and the column number must be given.

temperatures (2, 1) = 23
temperatures (4, 0) = 2

3.11 Declaring a two-dimensional array in VB.NET

Declaring a two-dimensional array:

Syntax 1:

```
{Dim | Private} arrayName(highestRowSubscript, highestColumnSubscript) _
                                                  As Datatype
```

Syntax 2:

```
{Dim | Private} arrayName(,) As Datatype = {Initial values}
```

Example 1:

```
Dim decSales(,) As Decimal = {{200.50, 130.99, 255.70}, _
                              {140.88, 123.50, 340.80}, _
                              {222.44, 111.23, 333.70}, _
                              {345.67, 889.21, 223.67}}
```

A two-dimensional array of 4 rows and 3 columns is declared. The elements will have the indicated initial values.

Further examples are:

decSales(0, 0) = 200.50
decSales(1, 2) = 340.80
decSales(2, 1) = 111.23 etc.

Example 2:

```
Dim decSales(5,6) As Decimal
```

An array with 6 rows and 7 columns will be declared. All elements will all have an initial value of 0.

3.12 Example of a two-dimensional array

There are 20 students in a class at the college. The names and student numbers of these students are stored in 2 parallel arrays in student number sequence.

Every student wrote 4 tests. These tests are stored in a two-dimensional array. Calculate the final mark for every student (the average of the four test marks) and store it in a third parallel array. Use the 3 parallel arrays to display all student numbers, followed by the student name in brackets and the final mark, on a list box. Determine the student who obtained the best mark and display the best mark and student name on a label.

For the purpose of this example all the data is available in the declarations.

Planning:

The following arrays will be used:

Student numbers	stNums	one dimensional	20 rows
Student names	stNames	one dimensional	20 rows
Student marks	stMarks	two dimensional	20 rows, 4 columns
Final Marks	stFinalMarks	one dimensional	20 rows

Other variables used:

The best mark	best
The total marks of a student	total
Variable for student loop	st
Variable for test loop	test

Algorithm:

CalcResults

```
~       The program will calculate the results of students

        for st = 0 to 19                        ~  loop student 20 times
                total = 0                       ~  initialise the total for 1 student
                for test = 0 to 3               ~  loop test 4 times
                        total = total + stMarks(st, test)
                                                ~  add a mark of a student to his/her total
                next test
                stFinalMark(st) = total / 4     ~  calculate and store average st mark in array
        next st

        for st = 0 to 19                        ~  display details of every student
                display "St no: ", stNum(st), "   St name:  ", stNames(st),
                                "   Final mark:  ", stFinalMark(st)
        next st
```

```
        ~  find the best student
        best = stFinalMark(0)                ~  set best to mark of 1st student
        for st = 0 to 19
                if stFinalMark(st) > best
                        best = stFinalMArk
                        student = stNames(st)        ~ store name as well
                endif
        next st
        ~  display the best student
        display "The best student is:  ", student, " and his mark is:  ", best
end
```

Code in the code editor:

```
Public Class CalcResults

Private Sub btnShow_Click(. . .) Handles btnShow.Click
   Dim strStNums() As String = {"209111222", "209123123", "209124124", _
                   "209126162", "209189234", "209223224", "209234243", _
                   "209267289", "209222333", "209333444", "209444555", _
                   "209555666", "209621721", "209732734", "209676787", _
                   "209777888", "209821822", "209845967", "208898999", _
                   "209921345"}

   Dim strStNames() As String = {"Danny McLean", "Sarah Cummings", _
                   "Mpho Ngwenya", "Gugu Nkosi", "Linda Watson", _
                   "Sue Peterson", "Harrold Ford", "Emmy Mokoena", _
                   "Sally Xaba", "Peter Mpshane", "Rob Smith", _
                   "Gwen Kabeni", "Larry Ramatshini", "Bob Boyd", _
                   "Ilse Ryan", "Jimmy Rammusi", "Precious Musi", _
                   "Kabelo Maake", "Tumelo Msiza", "Paul Ndamase"}

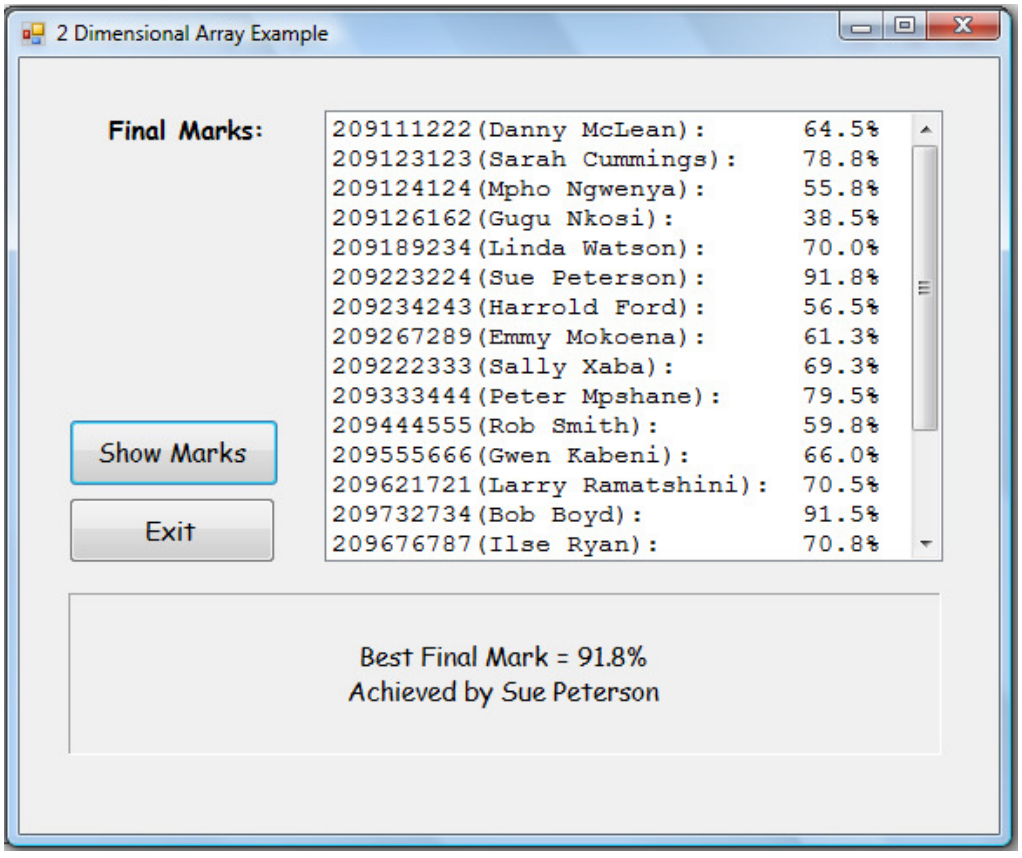
   Dim intStMarks(,) As Integer = {{67, 65, 64, 62}, _
                                   {76, 78, 79, 82}, _
                                   {55, 56, 58, 54}, _
                                   {43, 38, 41, 32}, _
                                   {56, 67, 78, 79}, _
                                   {88, 89, 95, 95}, _
                                   {65, 68, 44, 49}, _
                                   {63, 65, 59, 58}, _
                                   {67, 68, 69, 73}, _
                                   {77, 78, 82, 81}, _
                                   {67, 58, 49, 65}, _
                                   {66, 77, 66, 55}, _
                                   {69, 65, 73, 75}, _
                                   {91, 92, 94, 89}, _
                                   {77, 78, 69, 59}, _
                                   {33, 44, 55, 48}, _
                                   {55, 59, 61, 72}, _
                                   {77, 78, 65, 82}, _
                                   {84, 81, 78, 79}, _
                                   {66, 55, 77, 49}}

   Dim decStFinalMarks(19) As Decimal
   Dim decBest As Decimal
   Dim strStudent As String = ""
   Dim strDisplayStudent As String
   Dim intTotal As Integer = 0
   Dim st As Integer
```

```
    For st = 0 To 19
        intTotal = 0
        For test As Integer = 0 To 3
            intTotal = intTotal + intStMarks(st, test)
        Next
        decStFinalMarks(st) = Convert.ToDecimal(intTotal / 4)
    Next
    For st = 0 To 19
        strDisplayStudent = strStNums(st) & "(" & strStNames(st) & "):"
        lstMarks.Items.Add(strDisplayStudent.PadRight(30) & _
                           decStFinalMarks(st).ToString("N1") & "%")
    Next
    decBest = decStFinalMarks(0)
    For st = 1 To 19
        If decStFinalMarks(st) > decBest Then
           decBest = decStFinalMarks(st)
           strStudent = strStNames(st)
        End If
    Next
    lblAnswer.Text = "Best Final Mark = " & decBest.ToString("N1") & "%" & _
                     ControlChars.NewLine & "Achieved by " & strStudent
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
    Close()
End Sub
End Class
```

Output:





Practical

1. The BEST BUYS supermarket uses a two-dimensional array to store the department code, name and turnover of their 7 departments. At the end of every month, the amount sold during the month is added to the turnover. Plan and then write the VB.NET program to enter the 7 department numbers and amounts (in no specific order) and add them to the correct turnover amounts. At the end of the program the department numbers, department names and new turnover amounts must be displayed.

2. An array of characters contains a word of at least 10 characters in uppercase. Write a VB.NET program to remove all the vowels and move the rest of the elements in order not to have any spaces within the word e.g.

Original array:

D	I	M	E	N	S	I	O	N	A	L

Processed array:

D	M	N	S	N	L

Display the answer on the screen.

3.13 Sort an array using the bubble sort method

It is quite often advisable to sort the values of the elements in an array into a specific sequence - ascending or descending order according to a specific field(s). An example is the following: Suppose a programmer has to find a specific value in an array that has numeric elements. If the programmer is searching for the number in an unsorted array the possibility exists that all the elements have to be searched before the number is found or not found. But if the values of the elements are sorted in ascending numeric sequence, the programmer only searches until the number is found or until the next number in the array has a higher value than the number the programmer is searching for.

An unsorted array:

20	5	15	36	4

The same array sorted in descending sequence:

36	20	15	5	4

When using character or string values the order is:

Ascending: A to Z
Descending: Z to A

Remember that the names in a telephone directory are sorted in ascending sequence.

There are many sorting techniques. One of the simplest to understand is a technique called the **bubble sort**. Elements in the array are compared to each other in pairs, and when the element is not in sequence, it trades place with each other.

This concept will be explained by using an example as shown in the following array:

25	47	16	2	17

Comparing the first two elements: 25 is less than 47. Nothing changes

25	47	16	2	17

Comparing the next two elements: 47 is greater than 16. Values are swapped.

25	16	47	2	17

Comparing the next two elements: 47 is greater than 2. Values are swapped.

25	16	2	47	17

Comparing the next two elements: 47 is greater than 17. Values are swapped.

25	16	2	17	47

At this stage, you will notice that the highest value i.e. 47 is in the last position.

Now we are going to deal with the first 4 positions.

Compare the first two elements: 25 is greater than 16. Values are swapped.

16	25	2	17	47

Comparing the next two elements: 25 is greater than 2. Values are swapped.

16	2	25	17	47

Comparing the next two elements: 25 is greater than 17. Values are swapped.

16	2	17	25	47

At this stage, you will notice that the second highest value i.e. 25 is in the second last position.

Now we are going to deal with the first 3 positions.

Continue by comparing the first 3 position in the same manner; then the first 2 positions and the values in the array will be in ascending order.

3.14 EXAMPLE WITH A BUBBLE SORT

Problem:

The array called *petNames* has 15 elements that contain the names of pets. The values of the elements must be sorted in ascending sequence.

Algorithm:

SortPetNames

```
~       This program will sort the names of the pets in ascending order. We can assume that
~       the array already contains values.

        p = 12
        for k  =  0 to 13               ~  the array must be sorted 14 times
                for m = 0 to p          ~  number of comparisons
                        if petNames(m) > petNames(m+1)
                                temp = petNames(m)          ~  use temp to store the value
                                petNames(m) = petNames(m+1)
                                petNames(m+1) = temp
                        endif
                next m
                p = p – 1
        next k
end
```

Code in the code editor:

```
Public Class BubbleSort

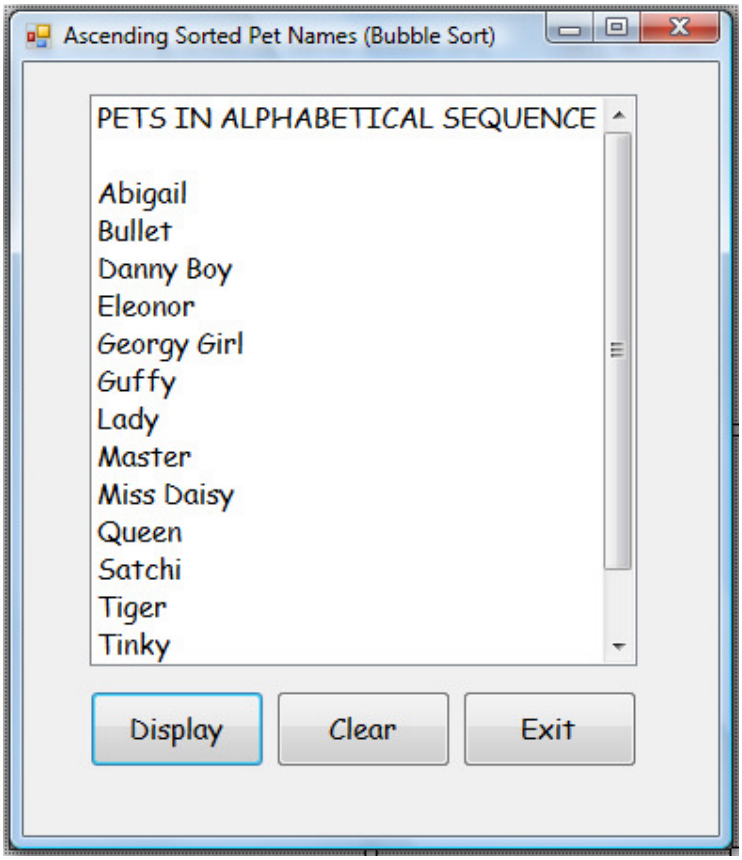
Private Sub btnDisplay_Click(. . .) Handles btnDisplay.Click
   Dim strPetNames() As String = {"Danny Boy", "Georgy Girl", "Satchi", _
       "Bullet", "Miss Daisy", "Guffy", "Lady", "Tinky", "Eleonor", _
       "Tiger", "Master", "Abigail", "Queen", "Trevor", "Snuffels"}
   Dim p, k, m As Integer
   p = 12
   For k = 0 To 13
       For m = 0 To p
           If strPetNames(m) > strPetNames(m + 1) Then
              Dim strTemp As String
              strTemp = strPetNames(m)
              strPetNames(m) = strPetNames(m + 1)
              strPetNames(m + 1) = strTemp
           End If
        Next m
        p = p - 1
    Next k
```

```
    lstDisplay.Items.Add("PETS IN ALPHABETICAL SEQUENCE")
    lstDisplay.Items.Add(" ")
    For p = 0 To 14
        lstDisplay.Items.Add(strPetNames(p))
    Next
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   lstDisplay.Items.Clear()
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
    Close()
End Sub
End Class
```

Output:

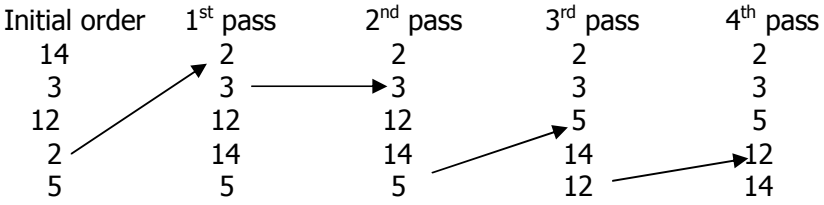


3.15 Sort an array using the selection sort method

Another sorting technique is called the **selection sort.** Using this method the smallest value is selected from the entire array to be exchanged with the first element in the array. Following this exchange, the second smallest element from the array is selected and exchanged with the second element. This process is continued until the entire array is in ascending sequence. To sort an array into descending sequence, the entire process is just

reversed to start with the biggest value and to exchange it with the first element in the array, etc.

Diagram for an ascending selection sort:



The arrows in the diagram indicate the movements of the smaller numbers. Note that they are exchanged with bigger numbers.

Let us sort the same array as in the previous example i.e. *petNames* that has 15 elements containing the names of pets. We are going to do the sort in **descending order**.

Note that (n-1) exchanges will take place if there are n elements in the array.

Algorithm

SortPetNames

```
~       This program will sort the names of the pets in descending order
~       The array already contains values

        for g = 0 to 13
                biggest = petNames(g)               ~  set initial values to
                position = g                        ~  find the biggest
                for h = (g+1) to 14
                        if petNames(h) > biggest
                                biggest = petNames(h)
                                position = h
                        endif
                next
                petNames(position) = petNames(g)
                petNames(g) = biggest
        next
End
```

Code in the code editor:

```
Public Class SelectionSort

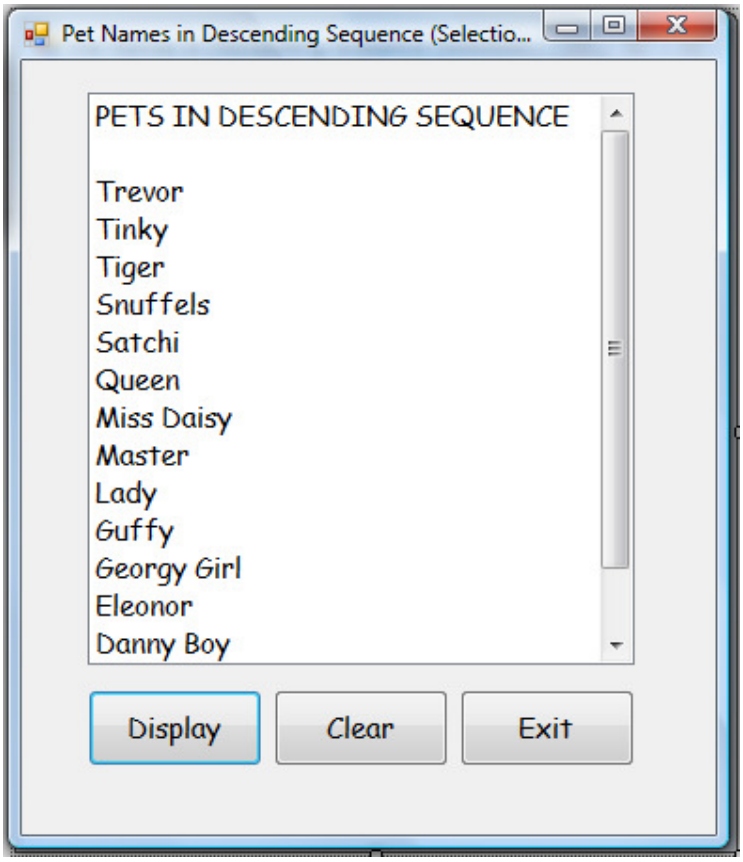
Private Sub btnDisplay_Click(. . .) Handles btnDisplay.Click
   Dim strPetNames() As String = {"Danny Boy", "Georgy Girl", "Satchi", _
       "Bullet", "Miss Daisy", "Guffy", "Lady", "Tinky", "Eleonor", _
       "Tiger", "Master", "Abigail", "Queen", "Trevor", "Snuffels"}
   Dim g, h As Integer
   Dim strBiggest As String
   Dim intPosition As Integer
```

```
    For g = 0 To 13
        strBiggest = strPetNames(g)
        intPosition = g
        For h = (g + 1) To 14
            If strPetNames(h) > strBiggest Then
               strBiggest = strPetNames(h)
               intPosition = h
            End If
        Next h
        strPetNames(intPosition) = strPetNames(g)
        strPetNames(g) = strBiggest
    Next g
    lstDisplay.Items.Add("PETS IN DESCENDING SEQUENCE")
    lstDisplay.Items.Add(" ")
    For g = 0 To 14
        lstDisplay.Items.Add(strPetNames(g))
    Next
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
    lstDisplay.Items.Clear()
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
    Close()
End Sub
End Class
```

Output:



3.16 Sort an array using VB.NET sort methods

You can use the Array.Sort Method to sort the elements in a one-dimensional array in ascending order and once sorted you can use the Array.Reverse Method to reverse the elements, i.e. sorting it in descending order.

Syntax of Array.Sort Method:

```
Array.Sort(ArrayName)
```

Syntax of Array.Reverse Method:

```
Array.Reverse(ArrayName)
```

In both cases, ArrayName is the name of a one dimensional array to be sorted.

Example 1:

```
Private Sub btnDisplay_Click(. . .) Handles btnDisplay.Click
   Dim strPetNames() As String = {"Danny Boy", "Georgy Girl", "Satchi", _
       "Bullet", "Miss Daisy", "Guffy", "Lady", "Tinky", "Eleonor", _
       "Tiger", "Master", "Abigail", "Queen", "Trevor", "Snuffels"}
   Array.Sort(strPetNames)
   lstDisplay.Items.Add("PETS IN ALPHABETICAL SEQUENCE")
   lstDisplay.Items.Add(" ")
   For p = 0 To 14
       lstDisplay.Items.Add(strPetNames(p))
   Next
End Sub
```

Example 2:

```
Private Sub btnDisplay_Click(. . .) Handles btnDisplay.Click
   Dim strPetNames() As String = {"Danny Boy", "Georgy Girl", "Satchi", _
       "Bullet", "Miss Daisy", "Guffy", "Lady", "Tinky", "Eleonor", _
       "Tiger", "Master", "Abigail", "Queen", "Trevor", "Snuffels"}
   Array.Sort(strPetNames)
   Array.Reverse(strPetNames)
   lstDisplay.Items.Add("PETS IN DESCENDING SEQUENCE")
   lstDisplay.Items.Add(" ")
   For p = 0 To 14
       lstDisplay.Items.Add(strPetNames(p))
   Next
End Sub
```

Although these methods seem much more convenient than the bubble sort and selection sort methods, it can only be used on one single array. If two parallel arrays must be sorted, the bubble sort or selection sort method will have to be used. E.g. one array contains student names and the other their respective test marks. If the array must be sorted in ascending sequence according to student names, the marks will have to be swapped when the names are swapped, otherwise every student could end up with another students mark!



Practical

1. Twenty athletes took part in a race and their times are stored in an array in minutes as real numbers. Use a bubble sort to sort the times (in ascending or descending order depending on your logic) and then display the times of the fastest (lowest) three athletes.

2. You may repeat the previous exercise by including a parallel array that contains the names of the athletes. Remember when doing the bubble sort, to swap the names in the names array at the same time as you swap times in the times array.

3. An array called numberArray contains an odd number of integer values. Use the selection sort to sort the array in ascending order and then display the lowest (first) value, the middle value and the biggest (last) integer. Enter the number of elements in the array in the beginning of the program as well as the values of the elements.

CHAPTER 4
Sequential Text Files

4.1 INTRODUCTION

A file is a collection of data stored together under a common name. Up to now, we have only used data that is entered on the keyboard and stored in temporary locations e.g. an array. Data in a file is usually stored on a more permanent medium e.g. a disk, a compact disk etc. The nature of the data stored is quite often also of a more permanent type e.g. the data of all the employees employed by a company.

There are different types of files, but in this course we will only discuss sequential access files. The organization of these files refers to the way data is stored on the file and the characters within the file are stored in a sequential manner, one after another. A sequential access file is sometimes also referred to as a text file.

The information in the file is stored in a sequential order and can only be accessed sequentially. For example the user will have to read the first 4 lines of text before the 5th line can be accessed.

OUTCOMES

When you have studied this chapter you should be able to:

- understand what a sequential file is
- understand the difference between input and output files
- do the following both in pseudo code an in VB.NET syntax:
    - open an input file to read text from it
    - open an output file to add text lines to it
    - open a new output file to create it with text lines
    - test if a sequential access file exists
    - close a sequential access file
    - write information to a sequential access file
    - read information from a sequential access file
    - test for the end of a sequential access file
- use the ReDim statement to change the number of elements in an array when the application is running
- create a structure
- declare and use a structure variable
- understand the advantages of using a structure
- create an array of structure variables and use it in sequential access file processing
- understand the basic concepts of databases.

4.2 INPUT AND OUTPUT FILES

The difference between input and output files is the way these files are used and accessed.

An **input file** has stored data that is used as input to a program when the data is **read** from the file into the program to be processed.

An **output file** is used by a program to store new data by **writing** it to the file. When dealing with sequential files a "new" file is always used because the changed or newly created data must be written or rewritten from the first to the last record.

It is not possible to write data to a specific position between other data on an input file, but more data may be added only at the end of the data on the input file. This is called appending of data.

Every file is identified by an external file name. Examples of file names are marks.txt and profit.txt.

4.3 OPENING A SEQUENTIAL ACCESS FILE

Every file that is used in the program must be opened before it can be used. The open statement that contains the name of the file, will try to find the file mentioned in the open statement. When the specific file is found, it will connect this external text file to the internal program file stream object in order to use it in the program.

As discussed in the previous paragraph there are input files, output files and output files to append data to. The programmer has to specify in the open statement what type of file must be opened.

In the generic programming (where we do the planning) the following open statements will be used:

Input file:

open input (name of file) e.g. open input (employees.txt)

The file pointer is now placed at the beginning of the file to start **reading** at the first line of text.

Output file:

There are two ways of opening an output file. It can either be opened to create a new output file or an existing file can be opened to append new text lines to it.

open outputCreate (name of file) e.g. open outputCreate (employees.txt)

This statement will create a new, empty sequential access file. The file pointer is now placed at the beginning of the file to start **writing** the first line of text.

open outputAppend (name of file) e.g. open outputAppend (employees.txt)

This statement will add new lines of text to an existing sequential access file. The file pointer will be placed at the end of the file. With every write statement, new lines of text will be **added** to the file.

If an input file does not exist, the program will end abnormally and therefore it is a good programming technique to first test to see whether the file exists before one tries to open it to read from it.

```
if exists (name of file)
        open the file
endif
```

A suitable action must normally be taken if the file does not exist.

Example:

```
if exists(students.txt)
        open input (students.txt)
else
        display "The file does not exist"
endif
```

4.4 CLOSING A SEQUENTIAL ACCESS FILE

A file should be closed as soon as possible to prevent losing information. In the case of an input file, it must be closed as soon as all the text has been read from it and when using an output file, as soon as all the text has been written to it.

In the case where quite a number of files are used at different stages in the program, it is very important to adhere to this rule by only keeping the files that are in use, open. It is also clear that files must be opened just before the data in it is needed.

The statement in the planning of a program to close a file is the following:

close (name of file)

Example:

close (employees.txt)

4.5 READING INFORMATION FROM A SEQUENTIAL ACCESS FILE

As mentioned earlier in this chapter, a sequential access file is always read from the beginning of the file. The open statement positions the pointer before the first character of text.

The **read statement** enables the program to read one line of text at a time, until the newline character is found. The string that is returned when the line is read does not include the newline character.

Lines can be read from a file until the end of file character is found that indicates that the file does not contain any data that has not been read.

The statement in an algorithm to read a line of text from a file is the following:

```
read (name of file)
```

The line of text is normally placed into a string variable from where it can be used for further processing.

Example:

```
EmployeeData = read (employees.txt)
```

In an algorithm, the word eof will indicate that the end of the file has been encountered and that processing must be stopped and the file be closed.

Example:

In this pseudo code example, lines of text are read from the employee file. Every line of text is then displayed. The process is repeated until the end of the file is reached.

```
if exists(employees.txt)
        open input (employees.txt)
        EmployeeData = read(employees.txt)
        Do until eof
                Display EmployeeData
                EmployeeData = read(employees.txt)
        Loop
        close (employees.txt)
else
        display "The file does not exist"
endif
```

Take note that the first line of text is read before the loop. It is displayed in the loop and before the next execution of the loop the next line of text must be read. If this next read statement is omitted, the first line of text will keep on being displayed because the next line is never read and the end of file will never been encountered. Every read statement will access the next line of text and will test if the end of the file has been reached.

4.6 WRITING INFORMATION TO A SEQUENTIAL ACCESS FILE

As mentioned earlier in this chapter, text lines can be added to an existing file or a new output file can be created. The open statement will indicate the way to write to the file.

In both cases the **write statement** enables the program to write one line of text at a time. The new line character will not automatically be written to the file and if the programmer wants the next string of text to be written on a new line, it must be specified.

Lines of text will normally also be written in a loop and when the end has been reached, the programmer can simply close the file.

The statement in an algorithm to write a line of text to a file is the following:

```
write (name of file)
```

The line of text is normally written from a string variable and in an algorithm this can be indicated by specifying the word from, followed by the variable name.

Example:

```
write (employees.txt) from EmployeeData
```

In an algorithm, an additional write statement must be included if the new line character must also be written. If it need not be written, this additional write statement can be ignored.

```
write new line
```

Example:

In this pseudo code example, employee data will be accepted from the keyboard and written to a new text file. When the user indicates that there are no more employees, the process will terminate and the file will be closed. Every line of employee data must be written to a new line on the file.

```
open outputCreate (employees.txt)   ~ Opens the file. If it exists, the current
                                    ~ content will be erased.
display "Enter the first line of employee data, enter -1 to stop"
enter EmployeeData
Do While EmployeeData <> "-1"
    write (employees.txt) from EmployeeData
    write new line
    display "Enter the next line of employee data, enter -1 to stop"
    enter EmployeeData
Loop
close (employees.txt)
```

Once again, the first line of text must be entered before the loop. It is written to the file in the loop and before the next execution of the loop, the next line of text must be entered. In this example, a value of -1 will indicate that the loop must be terminated and the file be closed.

4.7 INPUT FILES IN VB.NET

A sequence of characters is referred to as a stream of characters. In VB.NET, the **streamReader object** enables you to read data from a sequential file. Before this object is

created, a variable must first be declared to store this object in the internal memory of the computer. IO stands for Input-Ouput.

Syntax to declare a variable to store the streamReader object:

```
{Dim | Private} streamReaderVariableName As IO.StreamReader
```

Example:

```
Dim EmpInputFile As IO.StreamReader
```

The streamReader can now be created by using the **OpenText method** to open this sequential access file for input. By opening the file for input, it enables the computer to read the lines of text stored in the file.

Syntax of the OpenText Method:

```
streamReaderVariableName = IO.File.OpenText(filename)
```

Example:

```
EmpInputFile = IO.File.OpenText("C:\Employees.txt")
```

As mentioned in the generic planning part, it is always better to first test to see whether a file exists or not. This will avoid a runtime error if the computer tries to open a file that does not exist. The **Exists Method** can be used to test whether the file exists or not. This method returns a Boolean answer – true if the file exists, and false if the file does not exist.

Syntax of the Exists Method:

```
IO.File.Exists(filename)
```

Example:

```
If IO.File.Exists("C:\Employees.txt") = True Then
```

To read the contents of the text file line by line, the **ReadLine Method** can be used. A line is defined as a stream of characters (or a sequence of characters) followed by the newline character. Although the ReadLine method reads until the newline character is detected, it will only return the string. The newline character itself will not be returned. The string returned by the ReadLine method is normally assigned to a string variable from where data processing and manipulation can occur if necessary.

The ReadLine Method does not include a file name as it uses the file associated with the StreamReader variable.

Syntax of the ReadLine Method:

```
StringVariableName = StreamReaderVariableName.ReadLine
```

Example:

```
strEmployeeData = EmpInputFile.ReadLine
```

As the input file can contain many lines of text, it must be read line by line until the end of the file has been encountered. To achieve this, a loop structure can be used along with the **Peek Method** which "peeks" into the file to determine whether the file contains another character to read or not. If the file contains another character, the peek method will return the character otherwise it will return the number -1. The loop can therefore be repeated until the peek method returns a number -1.

Syntax of the Peek Method:

```
StreamReaderVariableName.Peek
```

Example:

```
strEmployeeData = EmpInputFile.ReadLine
Do Until EmpInputFile.Peek = -1
   lstDisplay.Items.Add(strEmployeeData)
   strEmployeeData = EmpInputFile.ReadLine
Loop
```

When all the lines of text have been read from the file and processed, the file can be closed by means of the **close method**.

Syntax of the Close Method:

```
StreamReaderVariableName.Close()
```

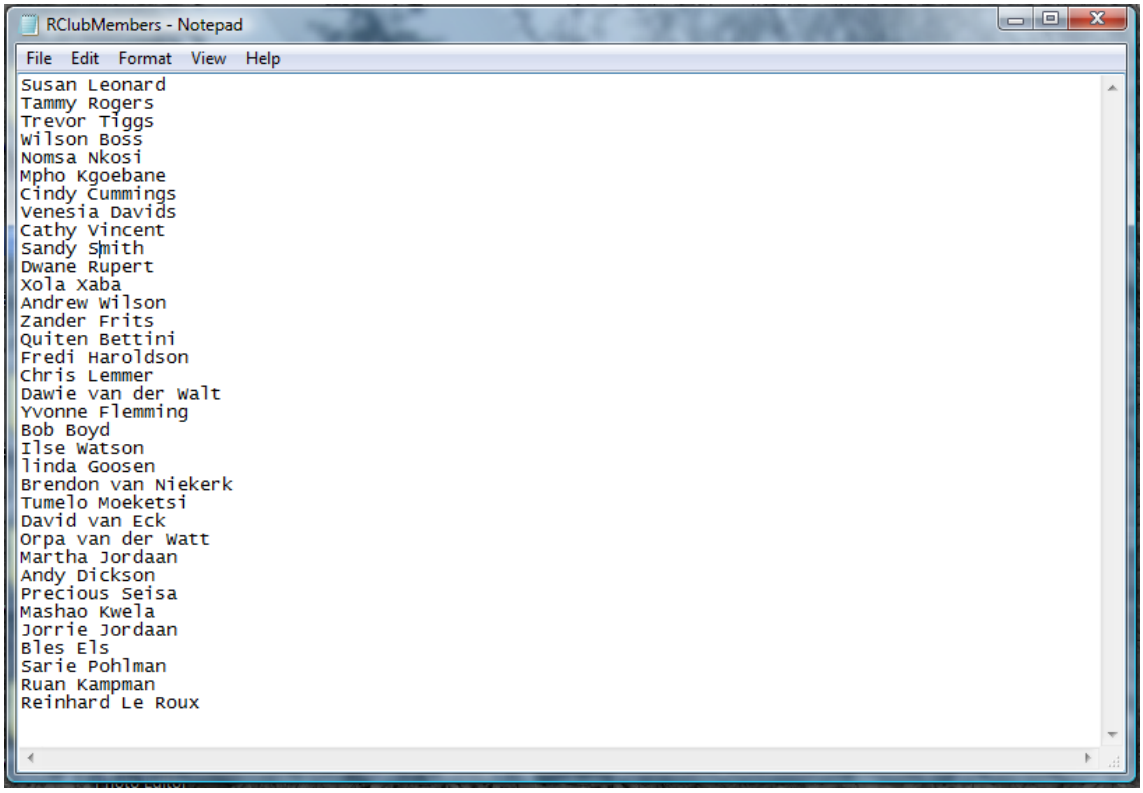
Example:

```
EmpInputFile.Close()
```

4.8 AN EXAMPLE WITH INPUT FILES

The RECREATION CLUB has a number of members that take part in different activities. The names of the members are stored on the file called RClubMembers.txt where the name in the first position will be that of the person who has been a member for the longest time and that of the last position will be the name of the person who joined the most recently.
The program must read these names from the file and display them in a list box.

Example of the sequential access file called RClubMembers.txt and stored on the C-drive:



Code in the code editor:

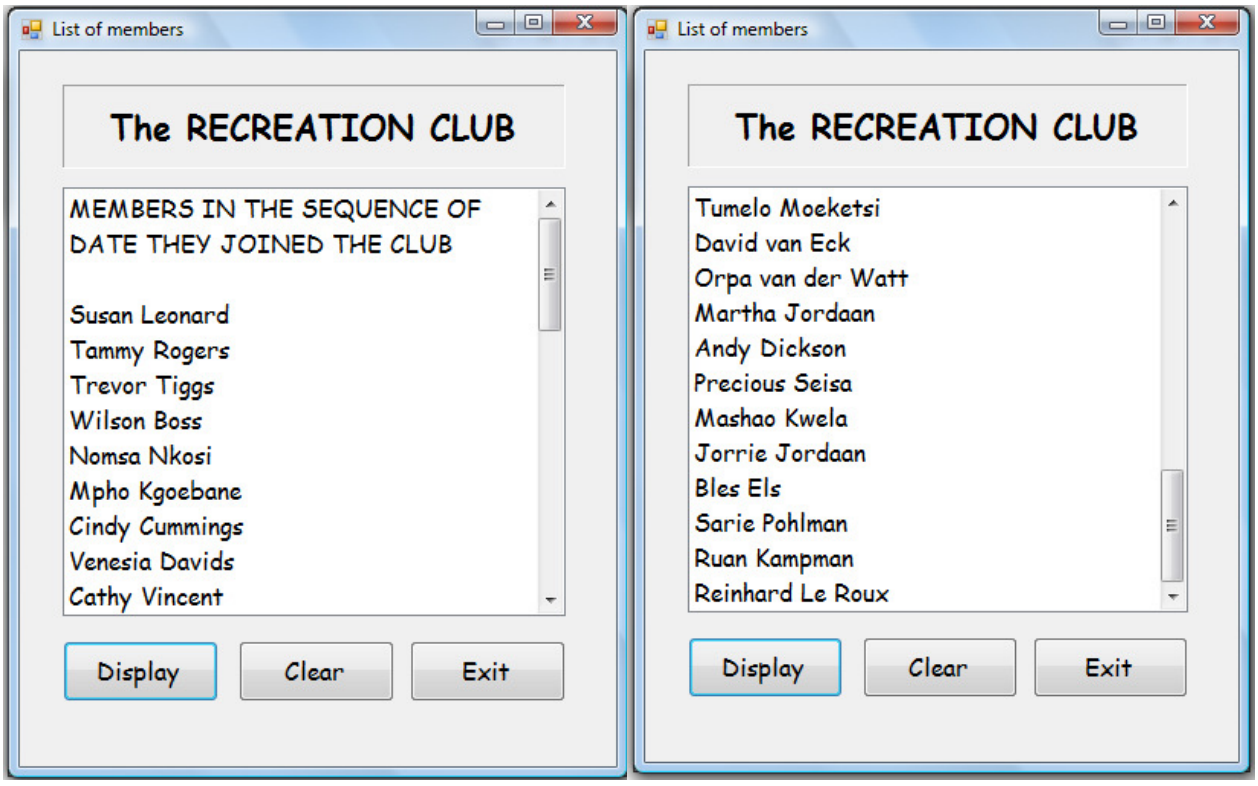
```
Public Class frmRClub

Private Sub btnDisplay_Click(. . .) Handles btnDisplay.Click
   Dim strMemberName As String
   Dim RClubFile As System.IO.StreamReader
   If System.IO.File.Exists("C:\RClubMembers.txt") Then
      lstMemberNames.Items.Add("MEMBERS IN THE SEQUENCE OF")
      lstMemberNames.Items.Add("DATE THEY JOINED THE CLUB")
      lstMemberNames.Items.Add(" ")
      RClubFile = System.IO.File.OpenText("C:\RClubMembers.txt")
      strMemberName = RClubFile.ReadLine()
      Do While RClubFile.Peek <> -1
         lstMemberNames.Items.Add(strMemberName)
         strMemberName = RClubFile.ReadLine()
      Loop
      RClubFile.Close()
   Else
      lstMemberNames.Items.Add("File with member names does not exist")
   End If
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   lstMemberNames.Items.Clear()
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Output produced by the program:



4.9 OUTPUT FILES IN VB.NET

The **streamWriter object** is used in VB.NET to write a stream of characters to a sequential access file. The programmer must declare a variable in which to store this object in the internal memory of the computer.

Syntax to declare a variable to store the streamWriter object:

```
{Dim | Private} streamWriterVariableName As IO.StreamWriter
```

Example:

```
Dim EmpOutputFile As IO.StreamWriter
```

After the variable has been declared, the programmer can create the streamWriter object. This involves opening the sequential access file in one of the following two ways:

CreateText

In this way, the sequential access file is opened for output. The computer creates a new, empty file, to which data can be written. If the file already exists, its contents will be erased before new data is written to it.

AppendText

The sequential access file is opened in order to append new data to it, i.e. new data is written to the file at the end of the existing data in the file. If the file does not exist, it will automatically be created.

In addition to opening the file, both these methods will automatically create the streamWriter object to represent the file in the application. The streamWriter object is assigned to the streamWriter variable in order to refer to the file in the code.

Syntax of the CreateText Method:

```
streamWriterVariableName = IO.File.CreateText(filename)
```

Example:

```
EmpOutputFile = IO.File.CreateText("C:\Employees.txt")
```

Syntax of the AppendText Method:

```
streamWriterVariableName = IO.File.AppendText(filename)
```

Example:

```
Dim strFileName As String = "C:\PayrollData.txt"
PayrollOutFile = IO.File.AppendText(strFileName)
```

After the file has been opened, data can be written to it by either using the **Write Method** or **WriteLine Method**. The WriteLine method writes a newline character at the end of the data. If every piece of data must be written on a new line, this method is the most likely method to use.

Syntax of the Write Method:

```
StringWriterVariableName.Write(data)
```

Example:

```
outputFile.Write("The name of my dog is ")
outputFile.Write("Danny Boy")
```

Result:

```
The name of my dog is Danny Boy
```

Syntax of the WriteLine Method:

```
StringWriterVariableName.WriteLine(data)
```

Example:

```
outputFile.WriteLine("The name of my dog is ")
outputFile.WriteLine("Danny Boy")
```

Result:

```
The name of my dog is
Danny Boy
```

When all the lines of text have been written to the file, the file can be closed by means of the **Close Method**.

Syntax of the Close Method:

```
StreamWriterVariableName.Close()
```

Example:

```
EmpOutputFile.Close()
```

4.10 AN EXAMPLE WITH OUTPUT FILES

A number of new members joined the RECREATION CLUB. Enter the names of these members in input boxes, display them on the list box and add them to the file called "C:\RClubMembers.txt".

Code in the code editor:

```
Public Class frmRClub

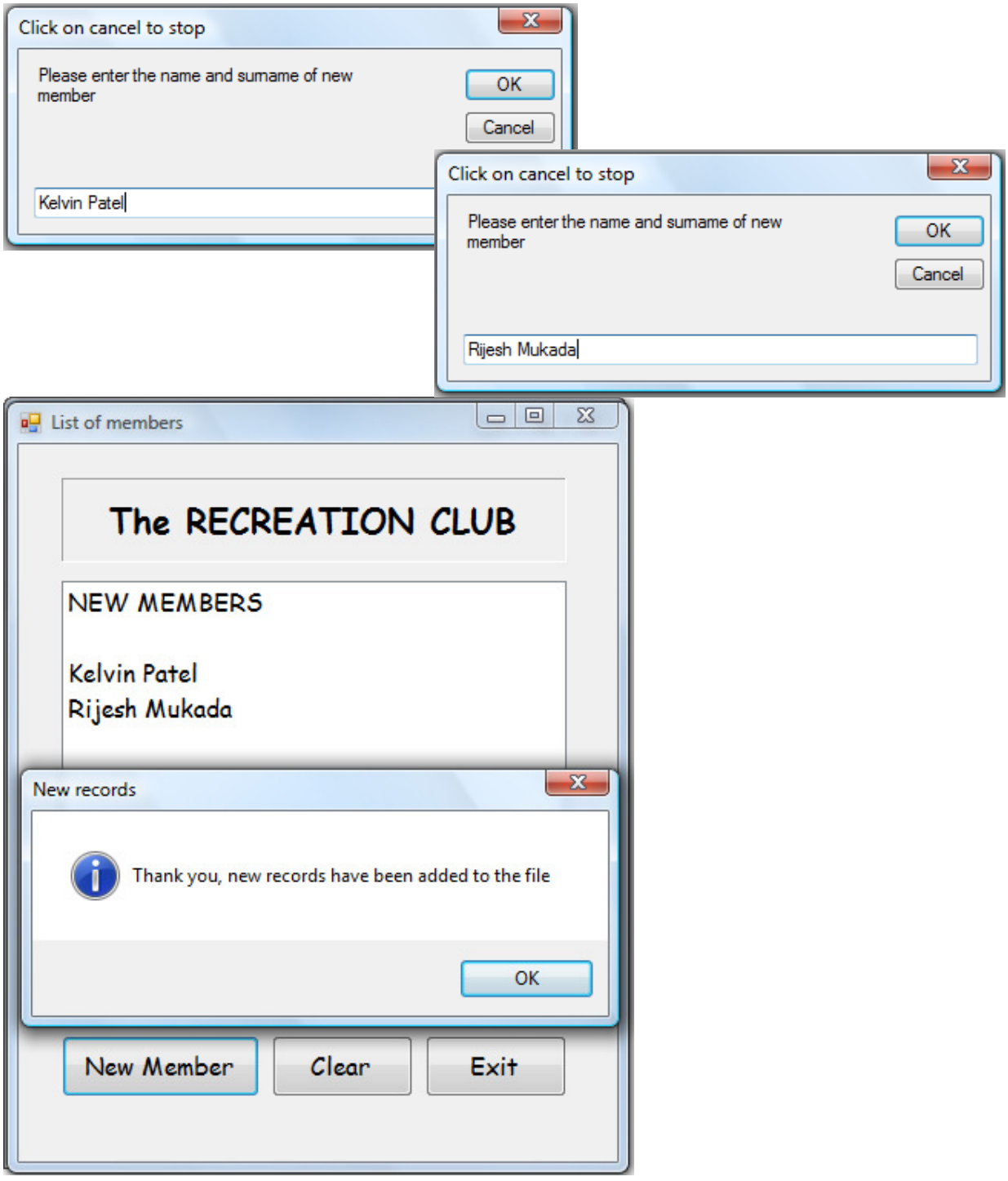
Private Sub btnNewMember_Click(. . .) Handles btnNewMember.Click
  Dim RClubFile As System.IO.StreamWriter
  Dim strNewMember As String
  RClubFile = System.IO.File.AppendText("C:\RClubMembers.txt")
  strNewMember = InputBox_
                 ("Please enter the name and surname of new member", _
                  "Click on cancel to stop")
  If strNewMember <> "" Then
    lstMemberNames.Items.Add("NEW MEMBERS")
    lstMemberNames.Items.Add(" ")
    Do While strNewMember <> ""
       lstMemberNames.Items.Add(strNewMember)
       RClubFile.WriteLine(strNewMember)
       strNewMember = InputBox_
                  ("Please enter the name and surname of new member", _
                  "Click on cancel to stop")
    Loop
    RClubFile.Close()
    MessageBox.Show("Thank you, new records have been added to the file", _
                "New records", MessageBoxButtons.OK, _
                 MessageBoxIcon.Information)

  Else
     lstMemberNames.Items.Add("No new members added")
  End If
End Sub

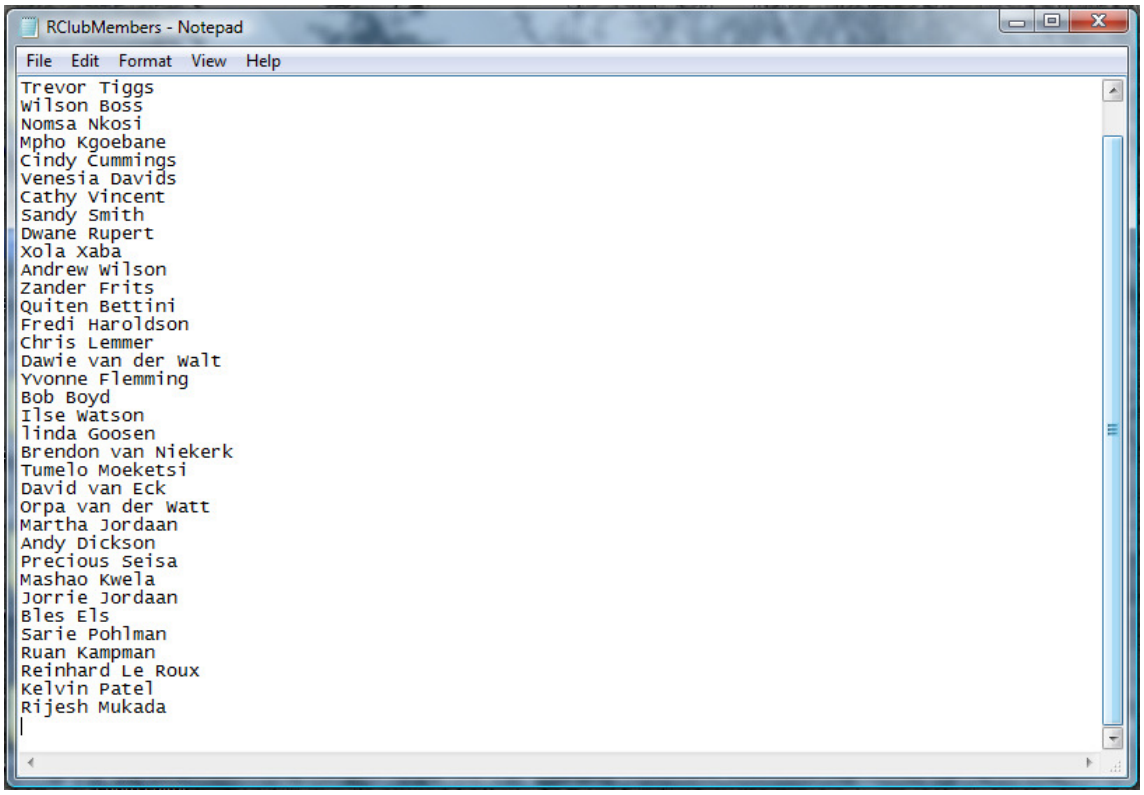
Private Sub btnClear_Click(. . .) Handles btnClear.Click
  lstMemberNames.Items.Clear()
End Sub
```

```
Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Examples of output:



Example of the file after the program has executed:



4.11 TO READ AND WRITE TO THE SAME SEQUENTIAL ACCESS FILE

If lines in a sequential file must be modified, the only way to achieve it would be to read all records into an array, manipulate the array elements and to write it back to the file by creating the file again. For example, if a member resigns from the RECREATION CLUB, the name of this member must be removed.

In a case like this, we do not know the exact number of elements in the array. If the number of members is initially 32, the array will have to be declared with 32 elements. However, if two members resigned, the size of the array must change to 30.

If the number of elements in an array can change while the application is running, a **ReDim** statement will be needed to either increase or decrease the number of elements in the array. If the original values in the array must be retained, the optional keyword, Preserve, must also be included in the statement otherwise all elements will be initialized.

Syntax of the ReDim statement:

```
ReDim [Preserve] arrayName(highestSubscript)
```

Example 1:

```
Dim intNumbers() As Integer = {6, 7, 8, 9}
ReDim Preserve intNumbers(5)
```

At first, the array called intNumbers will have a highest subscript of 3 and it will contain the following elements:

```
intNumbers(0) = 6
intNumbers(1) = 7
intNumbers(2) = 8
intNumbers(3) = 9
```

After the Redim statement, this array will have a highest subscript of 5 and because of the Preserve keyword it will retain its original values. However, 2 new elements will be added and their values would be initialized.

```
intNumbers(0) = 6
intNumbers(1) = 7
intNumbers(2) = 8
intNumbers(3) = 9
intNumbers(4) = 0
intNumbers(5) = 0
```

Example 2:

```
Dim intNumbers() As Integer = {6, 7}
ReDim intNumbers(3)
```

At first, the array called intNumbers will have a highest subscript of 1 and it will contain the following elements:

```
intNumbers(0) = 6
intNumbers(1) = 7
```

After the Redim statement, this array will have a highest subscript of 3, but all the elements will be initialized because no Preserve keyword occurred.

```
intNumbers(0) = 0
intNumbers(1) = 0
intNumbers(2) = 0
intNumbers(3) = 0
```

Example 3:

```
Dim intNumbers() As Integer = {6, 7, 8}
ReDim Preserve intNumbers(1)
```

At first, the array called intNumbers will have a highest subscript of 2 and it will contain the following elements:

```
intNumbers(0) = 6
intNumbers(1) = 7
intNumbers(2) = 8
```

After the Redim statement, this array will have a highest subscript of 1 and because of the Preserve keyword it will retain its original values.

```
intNumbers(0) = 6
intNumbers(1) = 7
```

In order to read and write to the same sequential access file, it will then be necessary to read all the lines of text to the elements in the array when the form is loaded. The original size of the array can be set to 0 and for every line to be stored in the array, the size of the array can be increased. When all elements have been read into the array, it can be manipulated when a button is clicked, and when the form closes down, all arrays elements can be written back to the file. In such a case all existing text in the file must first be erased and the file must be created from scratch.

4.12 EXAMPLE TO READ AND WRITE TO THE SAME SEQUENTIAL ACCESS FILE

Members at the RECREATION CLUB may resign. If a member resigns, the name of this member must be removed. We have to remember that the names are in a specific sequence and if the name of a member is removed, the names of all the other members must move to a higher position i.e. the previous number. E.g. if the name of the member in position 11 is removed, the name in position 12 moves to 11; 13 to 12 up to the end of the file.

Code in the code editor:

```
Public Class frmRClub

  'Declare an empty global array to contain all the lines of the text file.
  'The array is declared as global because the button click event, form
  'loading event and form closing event must all refer to it. All these
  'events must also refer to the indexes related to the array, therefore
  'the indexes are also declared as global variables. The boolean variable
  'is declared to test if the file initially exists or not. If not, the
  'other events may not be executed at all.

   Private strMemberName() As String = {}
   Private intArrayIndex As Integer = 0
   Private intRemoveIndex As Integer
   Private blnValidProcessing As Boolean = True

Private Sub frmRClub_Load(. . .) Handles MyBase.Load

  'When the form is loaded, the array must be loaded with lines of
  'text from the file (if it exists). You can code this procedure by double
  'clicking on the form.

   Dim InRClubFile As System.IO.StreamReader
   Dim strName As String

   If System.IO.File.Exists("C:\RClubMembers.txt") Then
      InRClubFile = System.IO.File.OpenText("C:\RClubMembers.txt")
      strName = InRClubFile.ReadLine
      Do While InRClubFile.Peek <> -1

         'Increase the size of the array by 1

         ReDim Preserve strMemberName(intArrayIndex)
         strMemberName(intArrayIndex) = strName
         lstMemberNames.Items.Add(strMemberName(intArrayIndex))
         strName = InRClubFile.ReadLine
         intArrayIndex += 1
      Loop
```

```
      'The array index will always be increased to prepare it for a
      'possible new line of text. When the end of file has been

      'reached, one must be deducted from it because there was no new
      'line of text (otherwise the array will have one empty element)

       intArrayIndex = intArrayIndex - 1
       InRClubFile.Close()
       MessageBox.Show("Array elements loaded. There are currently " & _
           intArrayIndex + 1 _
           & " members in the club, they will be displayed on the screen")
   Else

      'Set the boolean variable to false to indicate that the file does not
      'exist and that the other events may not be executed.

       blnValidProcessing = False
   End If
End Sub

Private Sub lstMemberNames_SelectedIndexChanged(. . .)Handles
lstMemberNames.SelectedIndexChanged

  'When the user selects a new item in the list box, the index must
  'be updated

   intRemoveIndex = lstMemberNames.SelectedIndex
End Sub

Private Sub btnRemove_Click(. . .) Handles btnRemove.Click
   Dim strNameRemove As String
   If blnValidProcessing = True

      'Test to see if the user has selected a member to be removed

      If lstMemberNames.SelectedIndex <> -1 Then
         strNameRemove = strMemberName(intRemoveIndex)
         For intNum As Integer = intRemoveIndex To intArrayIndex - 1
             strMemberName(intNum) = strMemberName(intNum + 1)
         Next

        'Update the size of the array - an element has been removed

         intArrayIndex = intArrayIndex - 1
         ReDim Preserve strMemberName(intArrayIndex)

         MessageBox.Show("Thank you, " & strNameRemove & _
             " has been removed as a member", " ", MessageBoxButtons.OK, _
             MessageBoxIcon.Information)

         'After the member has been removed from the array, the list box
         'must be cleared and the names must be displayed on the list box
         'again. If not, a second name can be selected and might reflect a
         'wrong position in the array due to the previous deletion and the
         'moving up of elements

         lstMemberNames.Items.Clear()
         For x As Integer = 0 To intArrayIndex
             lstMemberNames.Items.Add(strMemberName(x))
         Next
       Else
```

```
            MessageBox.Show( _
                 "You must first select a name of the member who resiged", _
                 "Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
         End If
      End If
End Sub

Private Sub frmRClub_FormClosed(. . .) Handles Me.FormClosed

  'When the form is closed, the array elements must be written back to the
  'file (only if the file initially existed)

   Dim OutRClubFile As System.IO.StreamWriter
   If blnValidProcessing = True
      OutRClubFile = System.IO.File.CreateText("C:\RClubMembers.txt")

     'Read all array elements and write it to the file

      For x As Integer = 0 To intArrayIndex
          OutRClubFile.WriteLine(strMemberName(x))
      Next
      OutRClubFile.Close()

      'Take note:
      'The following message is displayed on one line in the code editor
      'A line continuation character may not occur in the middle of a value

      MessageBox.Show("Thank you, new file with members created, there are
                currently " & intArrayIndex + 1 & " members in the club")
   End If
End Sub

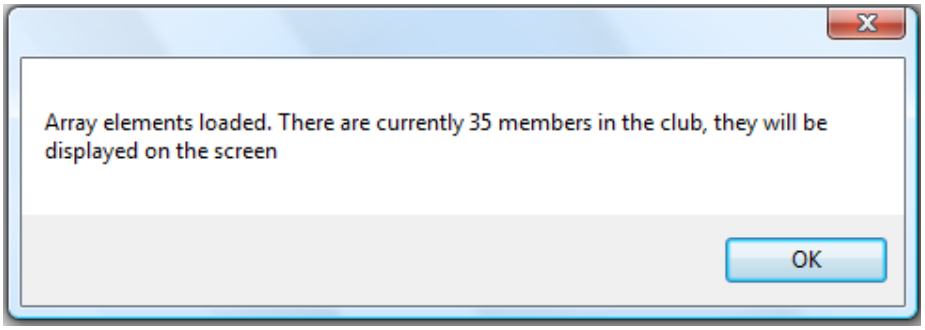
Private Sub btnClear_Click(. . .) Handles btnClear.Click
   lstMemberNames.Items.Clear()
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub

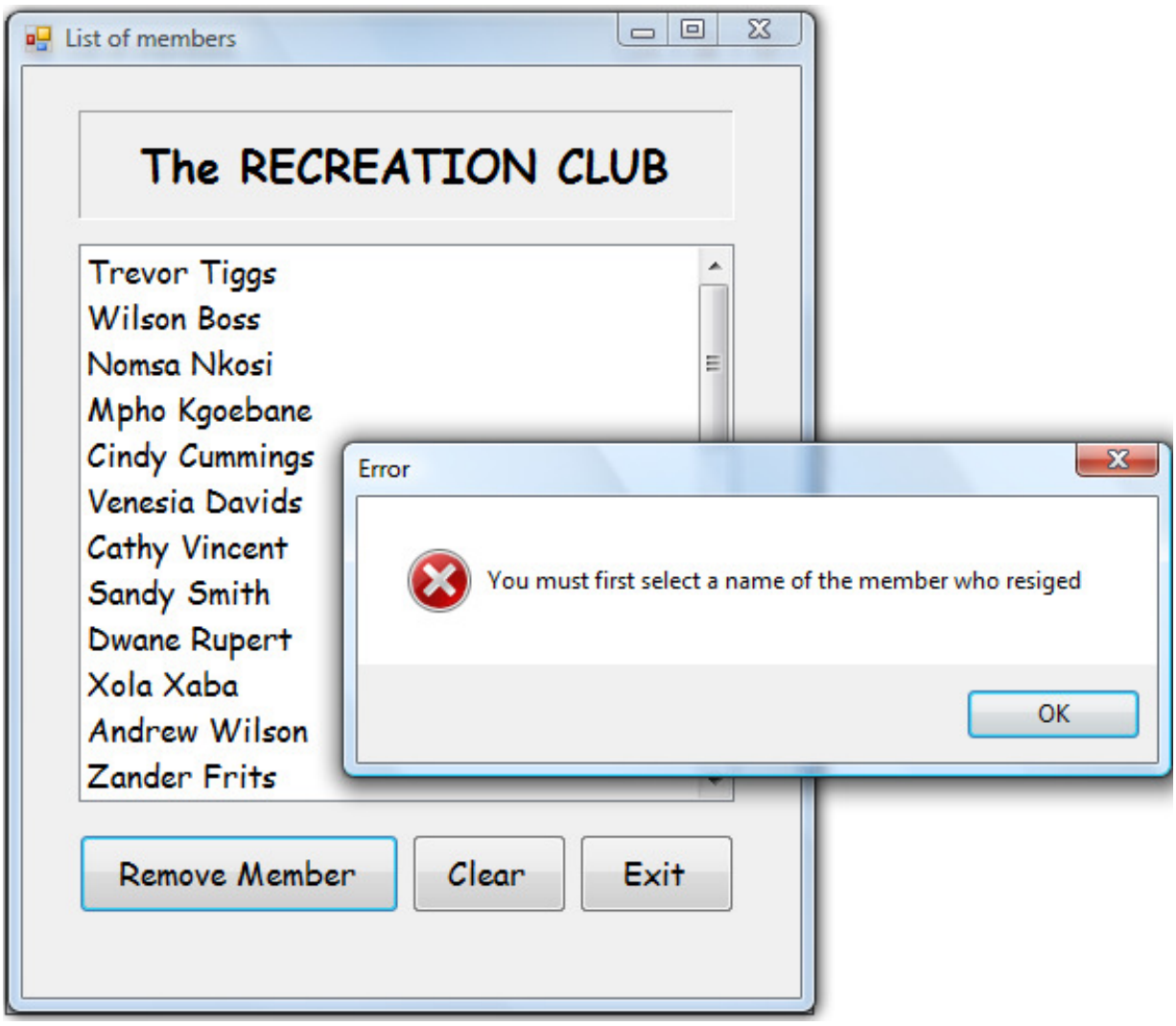
End Class
```

Output:

The following message box will be displayed before the member names are displayed on the list box.

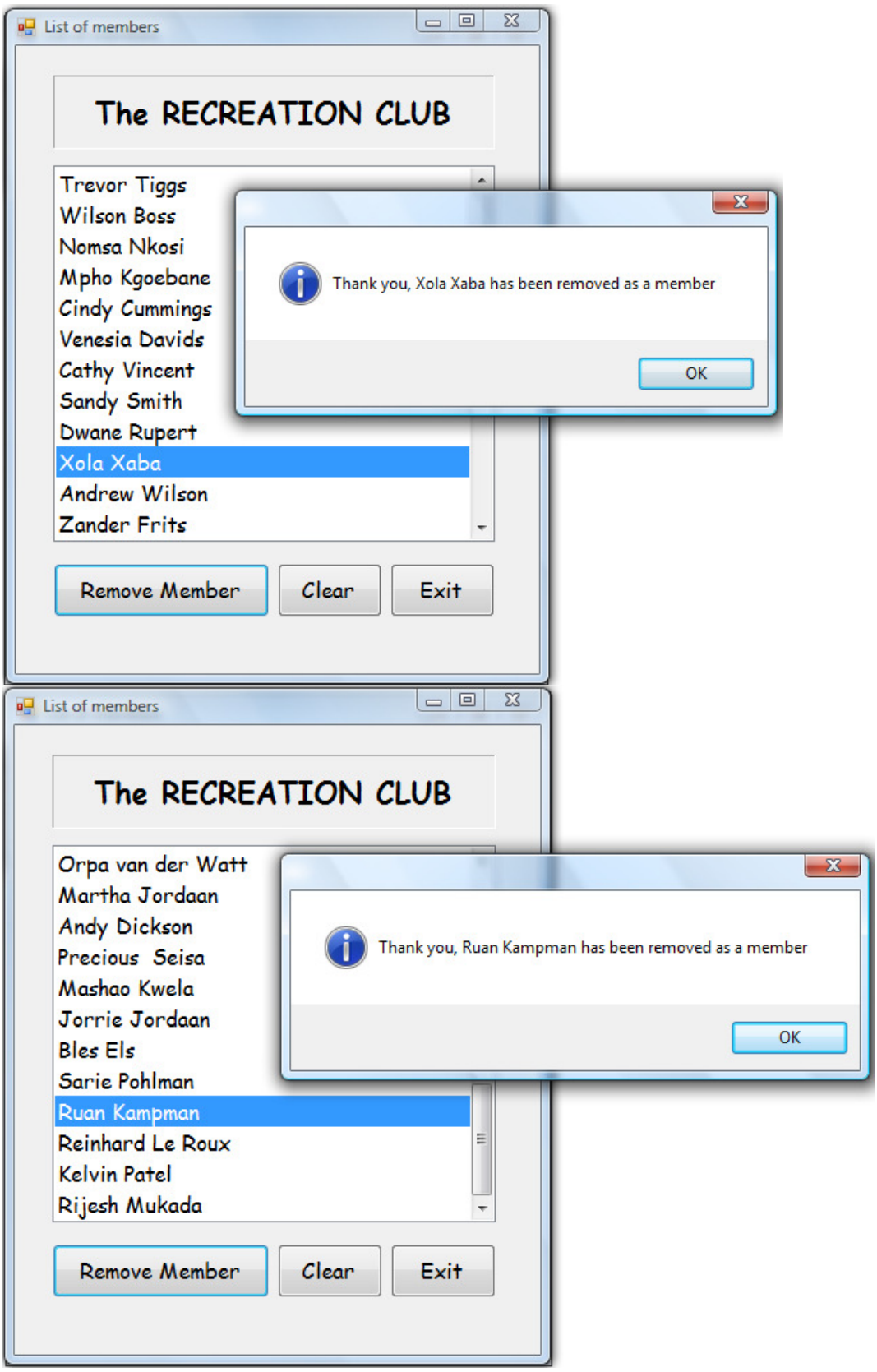


If the user clicks on the Remove Member-button before a name has been selected in the list box, the following error message will be displayed.

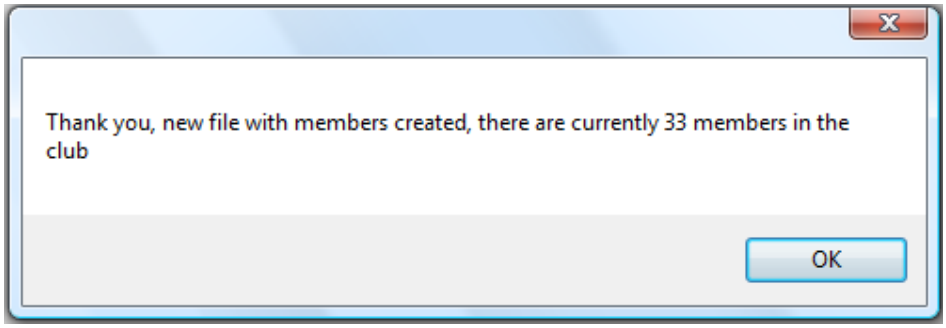


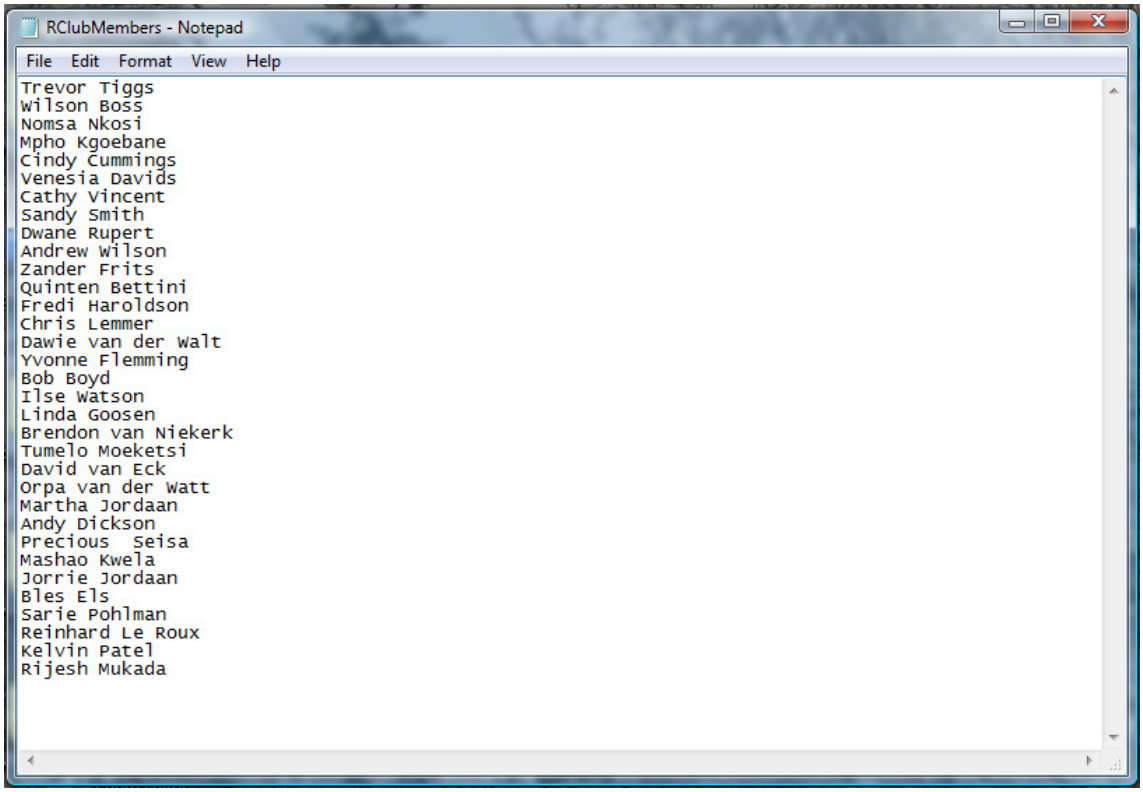
Take note that in this case, the list box must not contain headings as it will affect the index value of the line selected.

Assume that Xola Xaba as well as Ruan Kampman have resigned. The rest of the execution process will appear as follows.


List of members
The RECREATION CLUB
Trevor Tiggs
Wilson Boss
Nomsa Nkosi
Mpho Kgoebane
Cindy Cummings
Venesia Davids
Cathy Vincent
Sandy Smith
Dwane Rupert
Xola Xaba
Andrew Wilson
Zander Frits
Remove Member
Clear
Exit
Thank you, Xola Xaba has been removed as a member
OK
List of members
The RECREATION CLUB
Orpa van der Watt
Martha Jordaan
Andy Dickson
Precious Seisa
Mashao Kwela
Jorrie Jordaan
Bles Els
Sarie Pohlman
Ruan Kampman
Reinhard Le Roux
Kelvin Patel
Rijesh Mukada
Remove Member
Clear
Exit
Thank you, Ruan Kampman has been removed as a member
OK

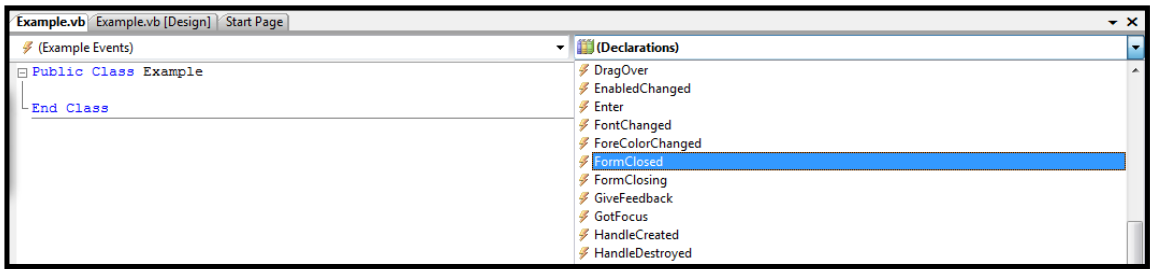
When the program is terminated, the following message box will be displayed to indicate that the file has been updated and that there are now 33 members in the club. The file C:\RClubMembers.txt now looks as follows:





Take note:

A form closing event can be included in the code by clicking on the declarations window in the code editor. Select FormClosed or FormClosing. The FormClosed event will be executed when the form is closed and the FormClosing just before it is closed. Both events could be used in order to write the array elements to the file.



4.13 CREATING A STRUCTURE IN VB.NET

In the previous file examples, every line of text in the file only contained one name and therefore it was quite easy to read from and to write to the file. However, applications are normally not that simple and every line in a file could contain a number of fields.

Example:
It could be possible to store the following fields for every employee who works in the company:

- Employee number
- Name
- Gender
- Department Code
- Annual Salary

If we now read one line of text, we must be able to divide it into all 5 these fields. In order to achieve this, we must use a structure.

A structure is a user-defined data type. Up to now we have used built-in Visual Basic data types (such as integer, string etc.) when declaring variables and arrays. You can also create your own data types in Visual Basic using the structure statement.

Syntax of a Structure:

```
Structure structureName
      Public memberVariable1 As Data Type
     [Public memberVariable2 As Data Type] . . .
End Structure
```

Between the structure and the end structure clauses, the members included in the structure are defined. The members can be variables, constants or procedures, but we will focus on variables.

Example:

```
Structure Employee
     Public strENumber As String
     Public strEName As String
     Public chaGender As Char
     Public chaDept As Char
     Public decAnnSalary As Decimal
End Structure
```

When a variable or array is declared in Visual Basic, it can be declared as the structure data type that you have defined.

Syntax to declare a Structure:

```
{Dim / Private} structureVariableName As StructureName
```

Examples:

```
'Declare the dean of the faculty as en employee
 Dim Dean As Employee

'Declare an array with 10 lecturers in the faculty of the data type
'Employee
 Dim Lecturers(9) As Employee
```

A programmer can refer to the entire structure variable by its name or to individual members by the structure name followed by a dot access operator (.) and the member name.

Examples:

```
Dean.strENumber = "1234545"
Dean.decAnnSalary = 350998.65D
Lecturers(4).chaGender = "F"
```

Structure members can also be used in calculations and can be manipulated in the normal way that you could do with variables, e.g.

```
Dean.decAnnSalary = Dean.decAnnSalary * 1.075D
```

Advantages of structures:

- A structure is also used to group related items. However, unlike an array, these items need not be of the same data type.
- Structures can be used to separate the individual fields in one text line in order to manipulate each one individually.
- In Visual Basic.NET, there is a restriction on the number of parameters that a function or procedure may receive. It will not be possible to send 10 value parameters to a function. However, if it is needed, it can be combined in a structure and a variable that has been declared of this structure type can be passed to the function.

4.14 AN EXAMPLE TO ILLUSTRATE SEQUENTIAL FILE PROCESSING USING STRUCTURES

The file C:\Employee.txt contains a text line with the following for every employee in a company.

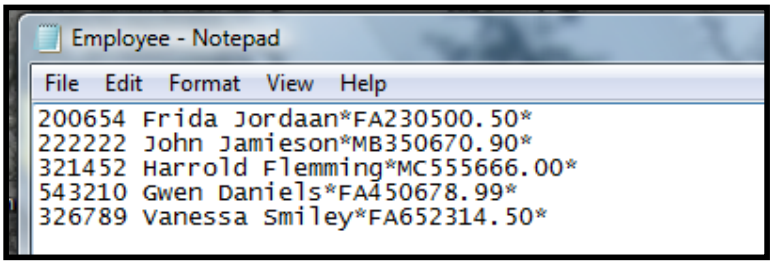
Employee number
Name
Gender
Department Code
Annual Salary

All the employees in department A received 10% increase, those in department B received 7.5% increase and all other employees received no increase. Update the file with these changes.

The gender will always be M or F and the department code will be only one character.

The programmer will have to use string manipulation methods in order to separate the individual fields for every employee. Therefore, it will be wise to insert a special character, such as a space, at the end of every field when the file is created. However, a space cannot be used to indicate the end of the name of the employee, because the name itself may contain a space. In such a case a different character such as an asterisk could be used. For fields such as the department and gender that are known to be only one character in length, an extra separation character would not be necessary.

Our input file could look as follows:



In this file, a space will indicate the end of the employee number, an asterisk will indicate the end of the employee name and the end of the annual salary. The gender and department code will both only be one character in length.

Code in the code editor:

```
Public Class Employees

    'Declaration of the structure

    Structure Employee
        Public strENumber As String
        Public strEName As String
        Public chaGender As Char
        Public chaDept As Char
        Public decAnnSal As Decimal
    End Structure

    'The array of the data type structure and the index are global because
    'more than one event will refer to it

    Private strEmployee() As Employee = {}
    Private intArrayIndex As Integer = 0

    'Declare a boolean variable to test if the file intially exists

    Private blnValidProcessing As Boolean = False
```

```
Private Sub Employees_Load(. . .) Handles MyBase.Load

   'When the form is loaded, the array must be loaded with lines of
   'text from the file. The array is a structure and string manipulation
   'methods will be used to get each member separately

    Dim InEmpFile As System.IO.StreamReader
    Dim strOneCompleteLine As String
    Dim intPosition As Integer
    If System.IO.File.Exists("C:\Employee.txt") Then
       InEmpFile = System.IO.File.OpenText("C:\Employee.txt")
       strOneCompleteLine = InEmpFile.ReadLine
       Do While InEmpFile.Peek <> -1

         'Increase the size of the array structure by 1

         ReDim Preserve strEmployee(intArrayIndex)

         'Determine the index of the space in the text line

         intPosition = strOneCompleteLine.IndexOf(" ")

         'Place all characters up to the space in the employee number
         'member of this array element

         strEmployee(intArrayIndex).strENumber = _
                     strOneCompleteLine.Substring(0, intPosition)

         'Remove the employee number and space from the string variable
         'The next substring method can then be applied from character 0
         'again.

         strOneCompleteLine = strOneCompleteLine.Remove(0, intPosition + 1)

          'Determine the index of the first asterisk in the text line

          intPosition = strOneCompleteLine.IndexOf("*")

         'Place all characters up to the asterisk in the employee name
         'member of this array element

         strEmployee(intArrayIndex).strEName = _
                         strOneCompleteLine.Substring(0, intPosition)

         'Remove the employee name and * from the string variable

         strOneCompleteLine = strOneCompleteLine.Remove(0, intPosition + 1)

         'Place one character into the gender member of the array element

         strEmployee(intArrayIndex).chaGender = _
                                    strOneCompleteLine.Substring(0, 1)

         'Remove the gender from the string variable

         strOneCompleteLine = strOneCompleteLine.Remove(0, 1)
```

```
            'Place the first character in the remaining string into the
            'department member of the array element

            strEmployee(intArrayIndex).chaDept = _
                                       strOneCompleteLine.Substring(0, 1)

            'Remove the Department from the string variable

            strOneCompleteLine = strOneCompleteLine.Remove(0, 1)

            'Find the position of the next asterisk (now the only remaining
            'asterisk in the string)

            intPosition = strOneCompleteLine.IndexOf("*")

            'Place all characters up to the asterisk in the annual salary
            'member of this array element

            strEmployee(intArrayIndex).decAnnSal = _
                         strOneCompleteLine.Substring(0, intPosition)

            'Read the next line of text

            strOneCompleteLine = InEmpFile.ReadLine

            'Increase the index to prepare it for another possible line of
            'text. In such a case, it will be placed in the next array
            'element. The size of the array will also be increased

            intArrayIndex += 1
         Loop

         'The array index will always be increased in the loop to prepare it
         'for a possible new line of text. When the end of file has been
         'reached 1 will have to be deducted from the index again because
         'there was no new line of text. If it was not deducted, the array
         'will have one empty element

         intArrayIndex = intArrayIndex - 1

         'Close the file, display a message to indicate that the array is
         'loaded and set the  boolean variable to true, to ensure that all
         'other events will execute

         InEmpFile.Close()
         MessageBox.Show("Array elements loaded")
         blnValidProcessing = True
      Else

         'If the file does not exist, display an error message and set a
         'boolean variable to ensure that no other event will occur.

         MessageBox.Show("Employee file does not exit, create it first")
         blnValidProcessing = False
      End If
   End Sub
```

```
Private Sub btnIncrease_Click(. . .) Handles btnIncrease.Click
    If blnValidProcessing = True Then
       lstDisplay.Items.Add("THE FOLLOWING EMPLOYEES RECEIVED AN INCREASE")
       lstDisplay.Items.Add("EMP NUMBER".PadRight(20) & _
                            "NEW ANNUAL SALARY".PadLeft(18))
       lstDisplay.Items.Add(" ")
       For x As Integer = 0 To intArrayIndex
           'For all array elements, test if the department is A or B in
           'which case an increase must be calculated. These values must
           'also be displayed in a list box
           If strEmployee(x).chaDept = "A" Then
              strEmployee(x).decAnnSal = strEmployee(x).decAnnSal * 1.1
              lstDisplay.Items.Add(strEmployee(x).strENumber.PadRight(20) _
                     & strEmployee(x).decAnnSal.ToString("N2").PadLeft(18))
           Else
              If strEmployee(x).chaDept = "B" Then
                 strEmployee(x).decAnnSal = _
                                  strEmployee(x).decAnnSal * 1.075
                 stDisplay.Items.Add( _
                   strEmployee(x).strENumber.PadRight(20) & _
                   strEmployee(x).decAnnSal.ToString("N2").PadLeft(18))
              End If
           End If
        Next
     Else
        MessageBox.Show("Sorry, you must create the file first")
     End If
End Sub

Private Sub Employees_FormClosing(. . .) Handles Me.FormClosing

    'When the form is closing, the array elements must be written back to
    'the file (only if the file existed in the first place)

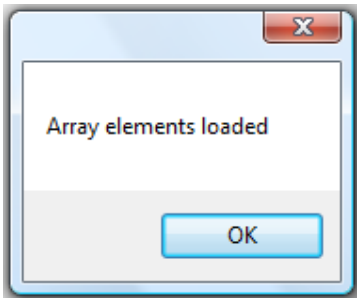
    If blnValidProcessing Then
       Dim OutEmpFile As System.IO.StreamWriter
       Dim strOutLine As String
       OutEmpFile = System.IO.File.CreateText("C:\Employee.txt")

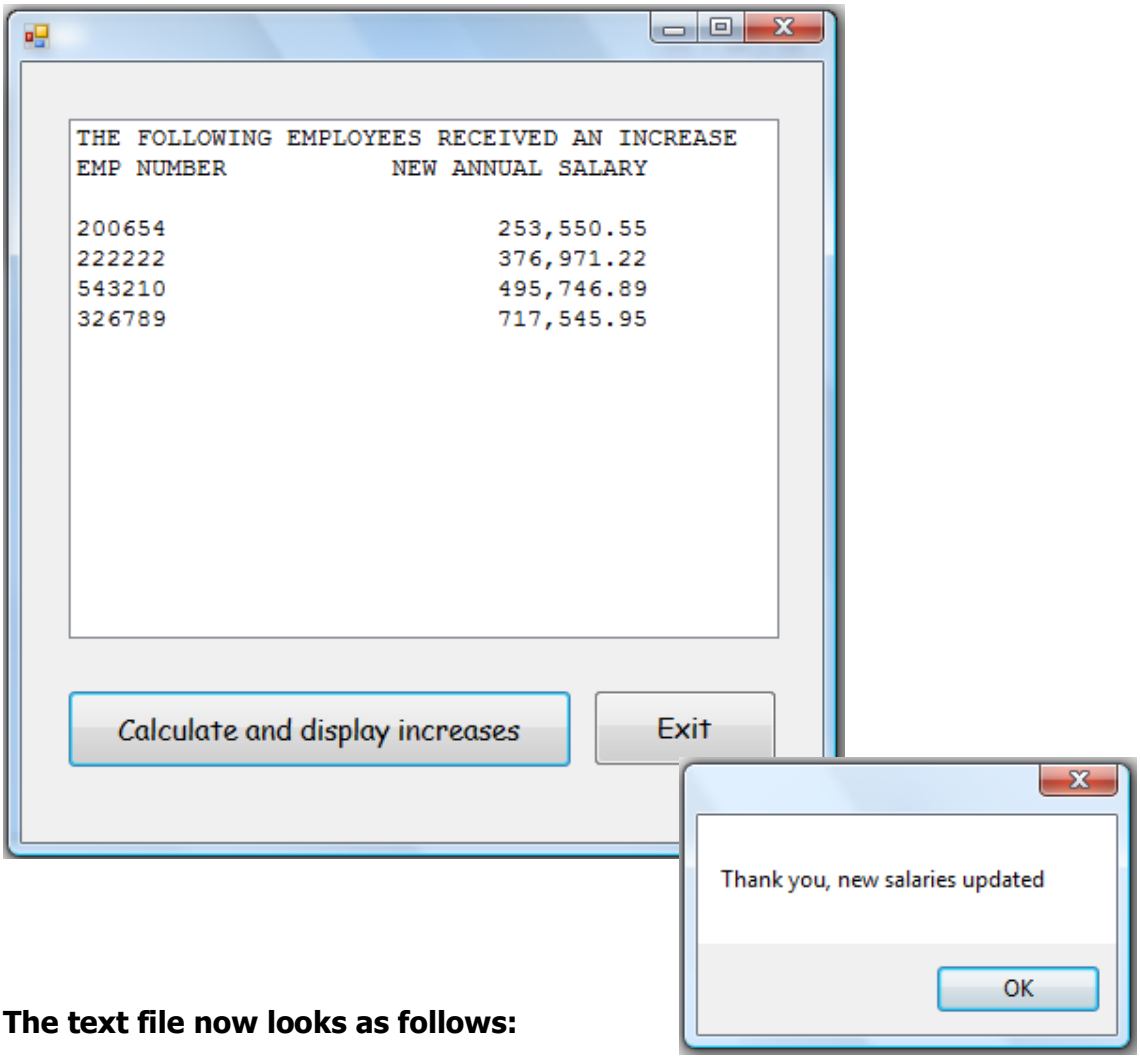
       'Concatenate all array elements into one string variable
       'Insert space and * where needed
       'Write this new string variable to the file

       For x As Integer = 0 To intArrayIndex
           strOutLine = strEmployee(x).strENumber & " " & _
                        strEmployee(x).strEName & "*" & _
                        strEmployee(x).chaGender & _
                        strEmployee(x).chaDept & _
                        strEmployee(x).decAnnSal.ToString("F2") & "*"
            OutEmpFile.WriteLine(strOutLine)
       Next
       OutEmpFile.Close()
       MessageBox.Show("Thank you, new salaries updated")
    End If
End Sub

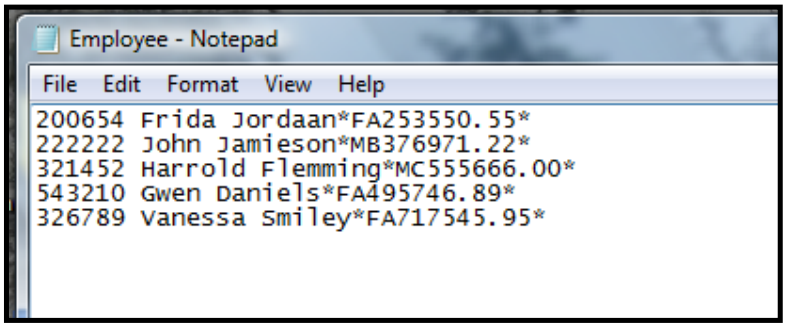
Private Sub btnExit_Click(. . .) Handles btnExit.Click
     Close()
End Sub
End Class
```

Output:





The text file now looks as follows:





Exercises

1. Create a structure to contain the following members:
   Student number
   Student name
   Predicate
   Exam
   Final Mark

2. Declare an array of this structure. There may be a maximum of 20 students.

3. Assign the following values to the fifth student in the array:

   Number = 20909091
   Name = T Malebane
   Predicate = 45
   Exam = 55
   Final Mark = 50



Practical

1. Do the necessary planning and write a VB.NET solution for the following:

   Dogs receive injections at 1ml per 10kg of the weight of the dog at the VET-FOR-YOUR-PET. An input sequential file contains a line per dog with the dog name (followed by an asterisk), the type of dog (followed by an asterisk) and the weight of the dog (followed by an asterisk). No dog may receive an injection of more than 10ml. Read all dogs into an array and then display the dog name and the ml injection to receive on a list box. Display suitable headings on the list box and display the output in columns.

2. Do the necessary planning and write a VB.NET solution for the following:

   Create a file with a line per student containing the following data for any number of students:
   Student number
   Student name
   Test Mark 1
   Test Mark 2
   Test Mark 3
   Only valid values may be accepted and all fields must be separated by #-signs on the file.

CHAPTER 5
Advanced Features and Built-In Functions

5.1 INTRODUCTION

In this chapter a number of useful and nice-to-have features and built-in functions will be discussed to help the programmer to write effective and useful programs.

OUTCOMES

When you have studied this chapter you should be able to:

- use methods in the math class
- understand and use the random number function
- understand and use the payment function
- be able to use the DateTimePicker in a program

5.2 METHODS IN THE MATH CLASS

The math class contains methods to assist the programmer to easily accomplish exponentiation, determining a square root, rounding of numbers, returning an absolute value as well as other mathematical methods.

5.2.1 Square Root

The method to calculate the square root of a number is the following:

Sqrt(number)

This method will return a value of the type double.

Example in pseudo code: answer = Sqrt(number)

Example in VB.NET: dblAnswer = Math.sqrt(intNumber)

5.2.2 Round

The method to round a number to a specific number of decimal positions is the following:

Round(number, positions) where positions represents the number of decimal positions

This method will return a value of the same type as the argument.

Example in pseudo code: amount = Round(amount, 2)

Example in VB.NET: decAmount = Math.Round(123.45678, 2)

5.2.3 Absolute value

The method to determine the absolute value of a number is the following:

Abs(number)

This method will return a value of the same type as the argument and will always be a positive number.

Example in pseudo code: distance = Abs(length)

Example in VB.NET: intDistance = Math.Abs(intLength)

5.2.4 Exponentiation

The method to raise a number to a specific power is the following:

Pow(number, p) where number is raised to the power of p.

This method will return a double value.

Example in pseudo code: numberSquare = Pow(number, 2)

Example in VB.NET: dblNumSquare = Math.Pow(intNumber, 2)

5.2.5 PI

The method allows you to use PI in a program without needing to include its numerical value of (3.1415926535897931).

This method will be illustrated by using it in a program to calculate and display the area of a circle of which the radius is 4 cm.

Algorithm Example:

```
CalculateArea
~       This program calculates and displays the area of the circle with a radius
~       of 4 cm.
        circleArea = Round(2 * PI * Pow(4, 2), 2)
        display "The area of the circle with radius 4 cm is ", circleArea
end
```

Example of the statement in VB.NET:

decCircleArea = Math.Round(2 * Math.PI * Math.Pow(4, 2), 2)

When you study the example above you have to note the following:

The first method to be processed is the PI method, followed by the Pow method. These two methods are within the inner brackets. Let us assume that the calculation within the inner bracket provides an answer called *result.* The equation can now be rewritten as:

circleArea = Round(result,2)

The Round method is the last method to be processed giving the circleArea with 2 decimal positions.

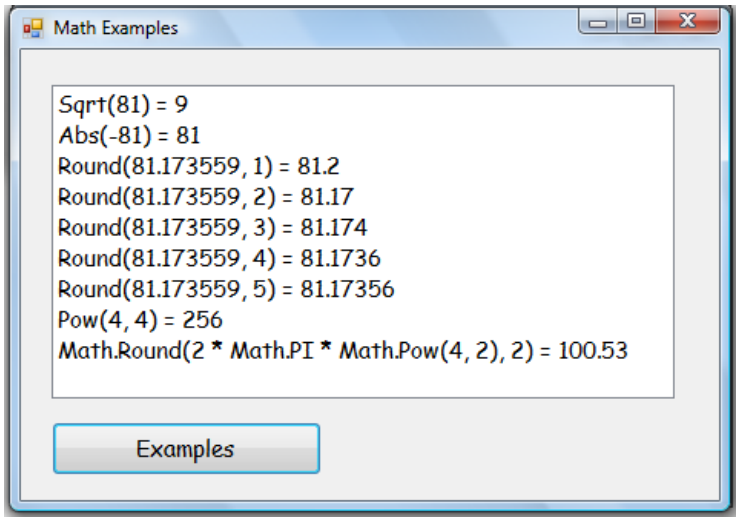
5.3 MATH EXAMPLES IN VB.NET

Code in the code editor:

```
Public Class MathExamples

Private Sub btnExamples_Click(. . .) Handles btnExamples.Click
    lstExamples.Items.Add("Sqrt(81) = " & Math.Sqrt(81))
    lstExamples.Items.Add("Abs(-81) = " & Math.Abs(-81))
    lstExamples.Items.Add("Round(81.173459, 1) = " & _
                          Math.Round(81.173559, 1))
    lstExamples.Items.Add("Round(81.173559, 2) = " & _
                          Math.Round(81.173559, 2))
        lstExamples.Items.Add("Round(81.173559, 3) = " & _
                          Math.Round(81.173559, 3))
        lstExamples.Items.Add("Round(81.173559, 4) = " & _
                          Math.Round(81.173559, 4))
        lstExamples.Items.Add("Round(81.173559, 5) = " & _
                          Math.Round(81.173559, 5))
        lstExamples.Items.Add("Pow(4, 4) = " & Math.Pow(4, 4))
        lstExamples.Items.Add(_
                  "Math.Round(2 * Math.PI * Math.Pow(4, 2), 2)" & _
                   Math.Round(2 * Math.PI * Math.Pow(4, 2), 2))
End Sub
End Class
```

Output:





Exercises

1. Calculate the value of each of the following:

   Sqrt(121)
   Round(-15.345,1)
   Abs(28)
   Round((PI * 2), 2)
   Pow(5,3)

2. The given variables have the following values:

   A = 12, B = 2, C = -2, D = 1.5, E = 121

Calculate the value of X in each of the following cases using the given variables. Show all the intermediate steps.

X = Pow(C,3)

X = Round ((Pow(3,B) – C),0)

X = Abs ((D * A) – Sqrt(A + 13) – A - 3)

X = Round ((Sqrt(E) – B / A), 2)

5.4 GENERATE RANDOM NUMBERS

The function to generate a random number enables the programmer to let "the computer" choose a number within a specific specified range. There are many opportunities to use random numbers e.g. in games, lottery, choosing people randomly by using an assigned number as well as in commercial and complicated statistical models.

VB.NET provides a pseudo-random generator that is a device to generate a sequence of numbers that meet certain statistical requirements for randomness. In order to use this generator, the programmer must first create a random object. This random object will represent the pseudo-random generator in the procedure where it is declared.

Syntax to create a random object:

```
{Dim | Private} randomObjectName As New Random
```

After the random object has been created, the random integers can be generated by using the Random.Next Method.

Syntax of Random.Next Method:

```
randomObjectName.Next(minValue, maxValue)
```

The arguments minValue and maxValue must be integers. The integer that will be generated will be greater than or equal to minValue and less than maxValue.

When the integer number is generated, it is normally stored as a variable of data type integer.

Example:

```
Dim RandomGenerator As New Random
intNumber = RandomGenerator.Next(1, 51)
```

In this case any number from 1 to 50 will be generated and stored in intNumber.

The statement to use in the planning of a program will be as follows:

number = random (1, 51)

The chosen number will be stored in the variable *number* and will contain a number from 1 up to and including 50.

5.5 RANDOM EXAMPLES

5.5.1 EXAMPLE 1

Do the planning and write the program to solve the following problems:

Obtain 50 random numbers from 20 to 200. Display all these numbers as well as the highest number that was generated.

	Description	Type	Name of variable
Input:	random numbers	integer	number
Output:	highest number	integer	highest

Algorithm:

```
Get Highest
        highest = 0
        for k = 1 to 50
                number = random (20, 201)
                display number
                if number > highest
                        highest = number
                endif
        next k
        display "The highest number generated was ", highest
end
```

VB.NET APPLICATION

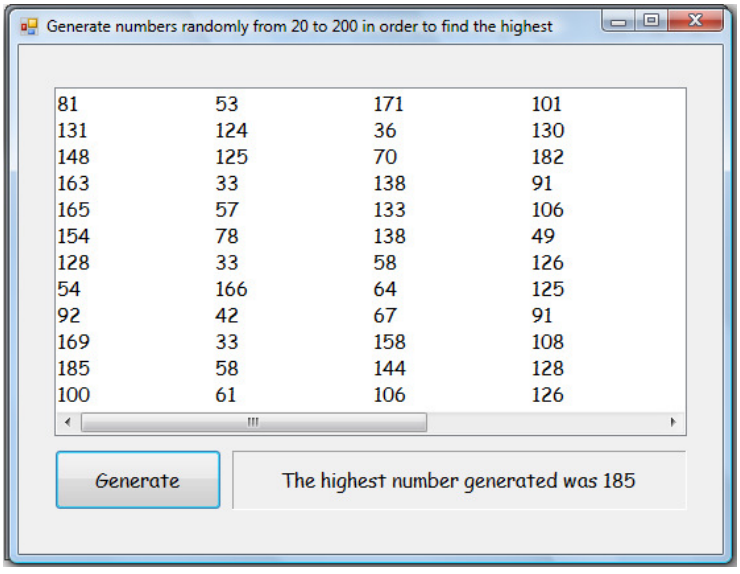
In this user interface, the numbers randomly generated will be displayed in a list box where the MultiColumn property has been set to TRUE.

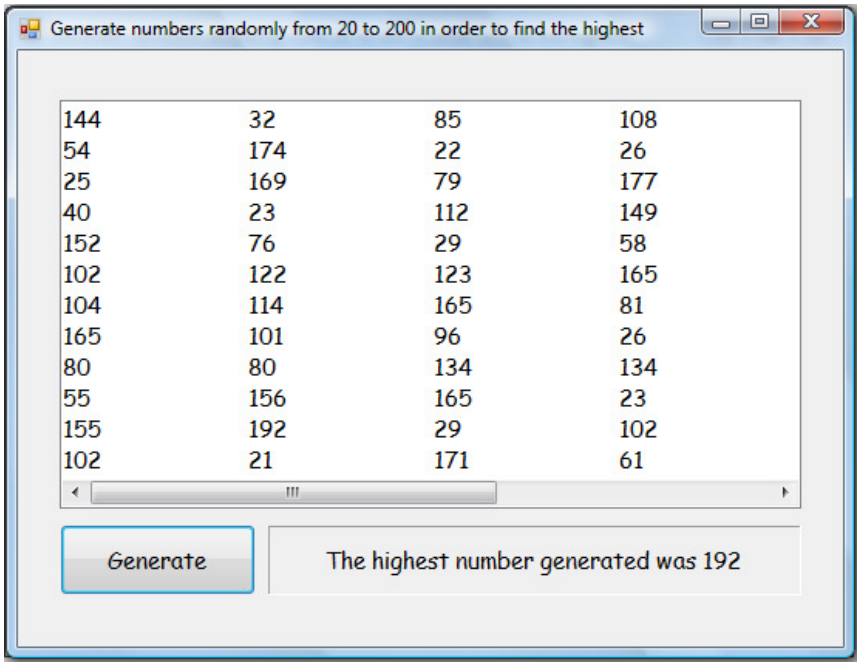
Code in the code editor:

```
Public Class GetHighest

Private Sub btnGenerate_Click(. . .) Handles btnGenerate.Click
   Dim RandomGenerator As New Random
   Dim intHighest As Integer = 0
   Dim intNumber As Integer
   For k As Integer = 1 To 50
       intNumber = RandomGenerator.Next(20, 201)
       lstDisplay.Items.Add(intNumber)
       If intNumber > intHighest Then
          intHighest = intNumber
       End If
   Next
   lblHighest.Text = "The highest number generated was " & intHighest
End Sub
End Class
```

Examples of output:





5.5.2 EXAMPLE 2

The user is asked to input any integer from 0 to 25. Obtain random numbers within the same range until the random number is equal to the number entered by the user. Display how many random numbers were generated before a match was found.

	Description	Type	Name of variable
Input:	random number	integer	randNumber
	user number	integer	userNumber
Output:	count of random numbers	integer	count

Algorithm:

```
CountEfforts
~       count the number of times a random number equal to the user's
~       number was entered.
        count = 1
        display "Choose any integer from 0 to 25"
        enter userNumber
        randNumber = random (0, 26)              ~  get first random number
        do while userNumber <>  randNumber
                randNumber = random (0, 26)
                count = count + 1
        enddo
        display "The number of random efforts before a match is ", count
end
```

VB.NET APPLICATION

Code in the code editor:

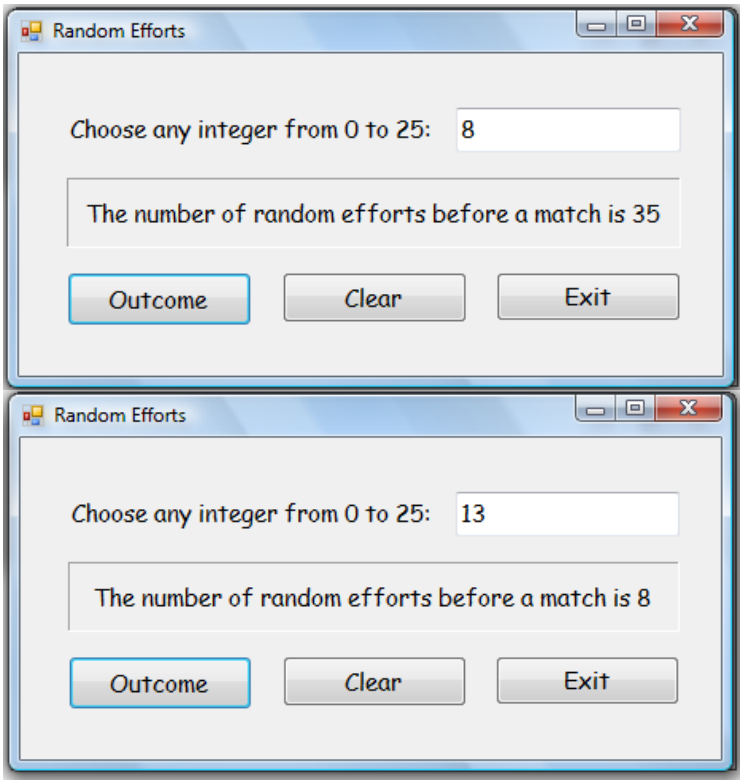
```
Public Class CountEfforts

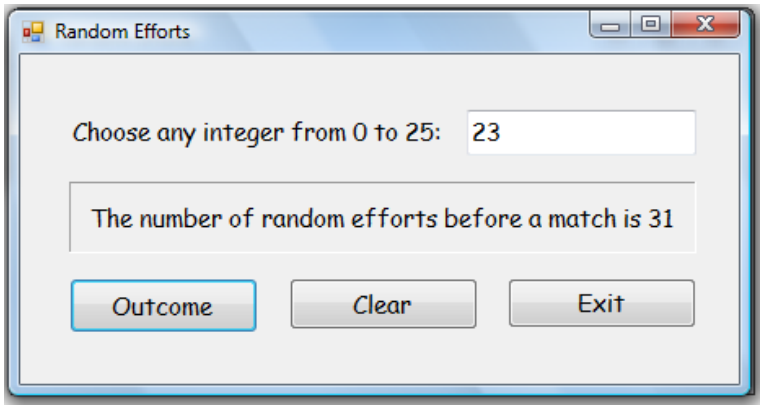
Private Sub btnOutcome_Click(. . .) Handles btnOutcome.Click
   'We will accept that the user enters a valid value
   Dim intCount = 1
   Dim intUserNumber, intRandomNumber As Integer
   Dim RandomGenerator As New Random
   intUserNumber = Convert.ToInt32(txtUserNumber.Text)
   intRandomNumber = RandomGenerator.Next(0, 26)
   Do While intUserNumber <> intRandomNumber
      intRandomNumber = RandomGenerator.Next(0, 26)
      intCount += 1
   Loop
   lblOutcome.Text = "The number of random efforts before a match is " _
                     & intCount.ToString
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
        txtUserNumber.Text = ""
        lblOutcome.Text = ""
        txtUserNumber.Focus()
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Examples of output:







Practical

When doing these exercises and processing the programs, your answers will never be the same as the random number are truly random and will provide you with different numbers every time.

Do the planning and write the programs for the following:

1. Use the Random.Next Method to choose a random number from 0 to 100. The program must continue getting random numbers until any number from 50 to 56 has been supplied. Terminate the program after the number of times a random number was produced is displayed.

2. Provide the minimum and maximum numbers of a range as well as the number of times a random number must be selected. Determine and display
   - what number was the lowest
   - the average of the random numbers
   - how many times a certain number was selected.

5.6 THE PMT METHOD

It happens quite often that a person needs to borrow an amount of money and wants to know beforehand how much the instalment that he has to pay, will be, when the rate of interest and number of instalments (period) are known. On the other hand it is also possible that a person wants to save a monthly amount of money and earn interest on the amount saved. Then the person will need to know how many months he will save a specific amount before his goal is reached.

We are going to use the Pmt method that will enable us to write a program to solve such a problem without hassles.

Syntax:

```
Financial.Pmt(Rate, NPer, PV [, FV, Due date])
```

Where

Pmt	-	the name of the function
Rate	-	interest rate per period (annually, monthly, etc)
NPer	-	total number of payment periods / investments (years, months, etc)
PV	-	Present value of loan or investment If loan, the loan amount / if investment, then 0
FV	-	Future value of loan or investment If loan, then 0 / if investment, then final amount to reach
Due date	-	Due date of payments Indicates if the amount is paid at the beginning or end of the period.

FV and Due date are optional. If they are omitted, a zero is assumed for the final amount (loan) and the end of the period (e.g. end of year if annually) is assumed for the date. The other arguments contain enough information to do the calculations.

In pseudo code we will simply use Pmt followed by the arguments in brackets as indicated in the following examples. The words "begin" or "end" will be used in planning for the due date.

Take Note:

The Financial.Pmt Method returns the answer as a negative value. If it must be displayed as a positive value, the abs-function will first have to be applied to it.

5.6.1 Short PMT Examples:

1. Samson borrowed R2000. He pays an annual interest rate of 11% and pays 4 annual instalments at the end of every year. How much is every instalment?

   Instalment = Pmt(0.11, 4, 2000, 0, end)
   or just
   Instalment = Pmt(0.11, 4, 2000) because the last two arguments are optional.

2. Alfred borrowed R3500. He pays an annual interest rate of 11.5% and pays monthly instalments at the end of every month for 3 years. How much is every instalment.

   We must be very careful now, because of the "monthly" factor. The interest rate will now change to 11.5% / 12 and the number of instalments to 3 * 12.

   Instalment = Pmt(0.115/12, 3 * 12, 3500)

3. Casandra wants to buy a pair of boots before the next winter and needs R1050. She will be saving for 10 months. She will earn 7.5% interest per annum. How much will she have to save at the beginning of every month?

Instalment = Pmt(0.075/12, 10, 0, 1050, begin)

5.7 PMT EXAMPLE

Do the planning and write a program to solve the following problem:

Frank and Wendy want to buy a house and a car. But they decided to buy the house and pay the monthly instalments over a period of 25 years. The bank offered them a loan at an interest rate of 12% per annum. They are not going to buy the car at the same time, but will start saving an amount at the end of every 3 months for the next 5 years. The bank offered them an interest rate of 8% per annum on their savings. The user has to input the price of the house as well as the amount of money needed for the car. Display the answers on the screen.

	Description	Type	Name of variable
Input:	price of house	float	housePrice
	Amount to save for car	float	carPrice
Output:	instalment on house loan	float	houseInstalment
	instalment on savings	float	carInstalment

Algorithm:

CalculateInstalments

display "Provide the price of the house"
enter housePrice

display "Provide the price of the car"
enter carPrice

houseInstalment = Abs(Pmt(0.12/12, 25*12, housePrice))

carInstalment = Abs(Pmt(0.08/4, 5*4, 0, carPrice)) ~ 4 instalments per year

display "Frank and Wendy pay R", houseInstalment, " per month for their house"
display " and have to save R", carInstalment, " every 3 months"

end

Code in the code editor:

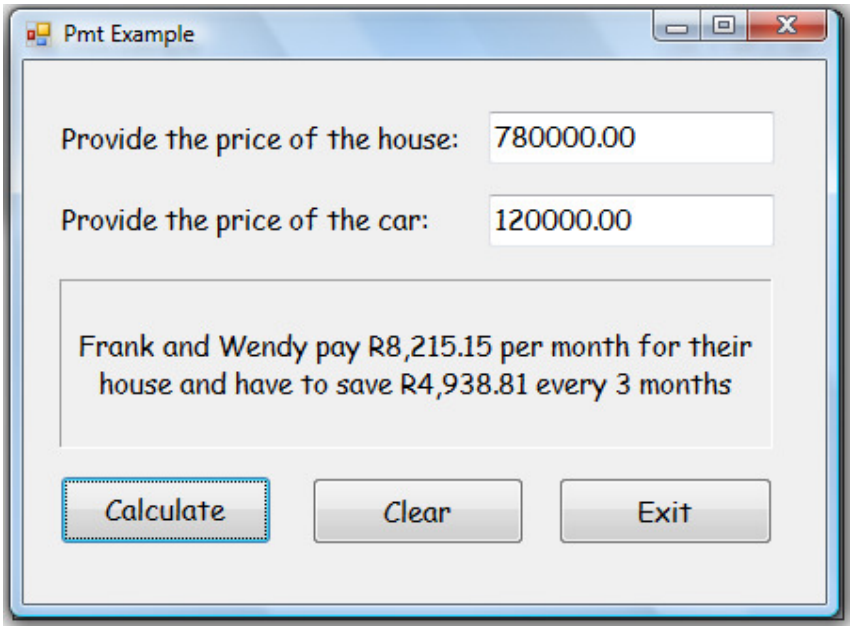
```
Public Class CalcInstalments

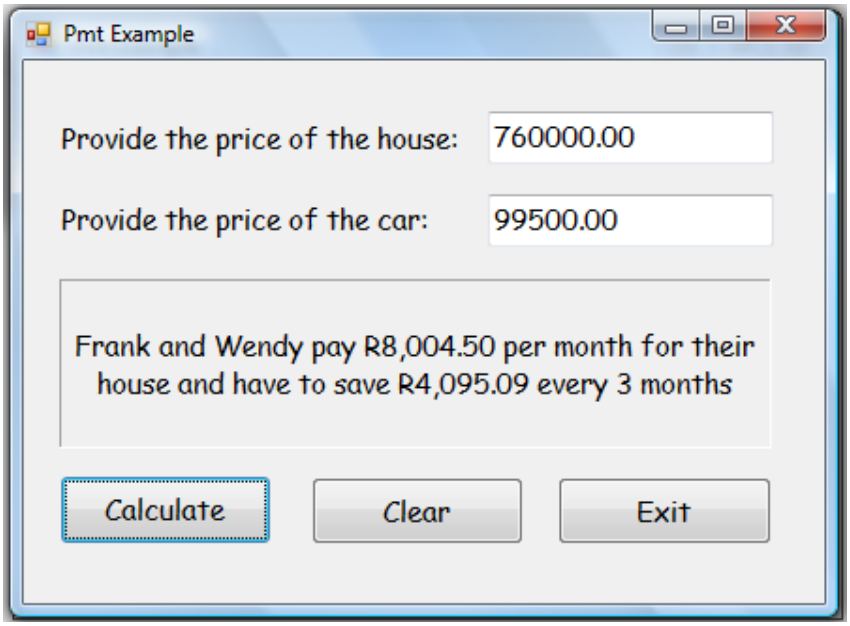
Private Sub btnCalc_Click(. . .) Handles btnCalc.Click
   Dim decHousePrice, decCarPrice As Decimal
   Dim decHouseInstalment, decCarInstalment As Decimal
   Decimal.TryParse(txtHousePrice.Text, decHousePrice)
   Decimal.TryParse(txtCarPrice.Text, decCarPrice)
   decHouseInstalment = Math.Abs(Financial.Pmt(0.12 / 12, _
                        25 * 12, decHousePrice))
   decCarInstalment = Math.Abs(Financial.Pmt(0.08 / 4, 5 * 4, _
                        0, decCarPrice))
   lblInstalment.Text = "Frank and Wendy pay R" & _
             decHouseInstalment.ToString("N2") & _
             " per month for their house and have to save R" & _
             decCarInstalment.ToString("N2") & " every 3 months"
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   txtHousePrice.Text = ""
   txtCarPrice.Text = ""
   lblInstalment.Text = ""
   txtHousePrice.Focus()
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

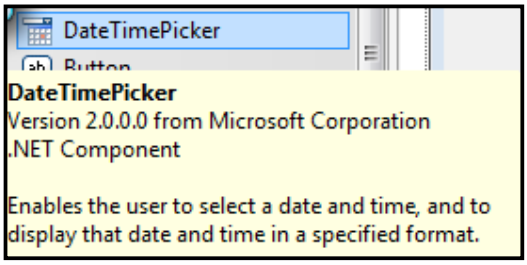
Possible output:





5.8 ADDING A DATE-TIME-PICKER CONTROL TO A FORM

The **DateTimePicker tool**, that is found in the toolbox, enables the programmer to include a date on a label in a form.



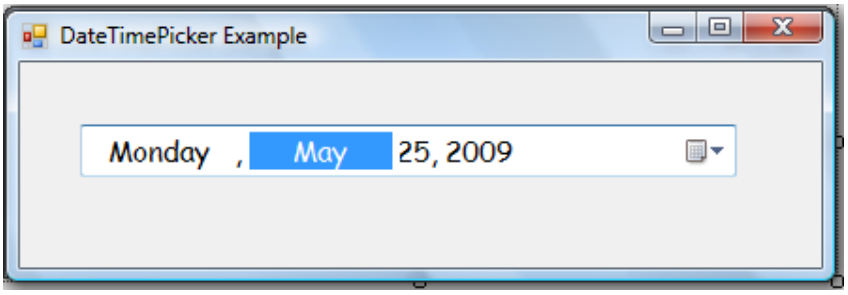
The DateTimePicker can include today's date (the system date) on the form or is able to display a calendar from which of date may be picked.

The format property can be used to control the format of the date or the time displayed in the DateTimePicker control as can be seen in the following table:

Setting	**Purpose**	**Examples**
Long	Displays the date using the long format	Thursday , May 07, 2009 Wednesday, September 23, 2009
Short	Displays a date using the short format	7/05/2009 23/09/2009
Time	Displays the time using the format h:mm:ss tt Note that AM and PM are used and 0 – 12 hours not 0 to 24 hours	5:15:45 PM 10:30:02 AM

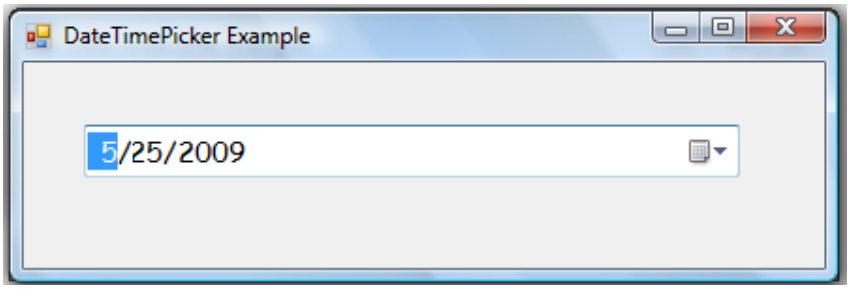
Example 1:

In this example, the DateTimePicker has been included on the form, and the format property of the control is Long. The current date is 25 May 2009.



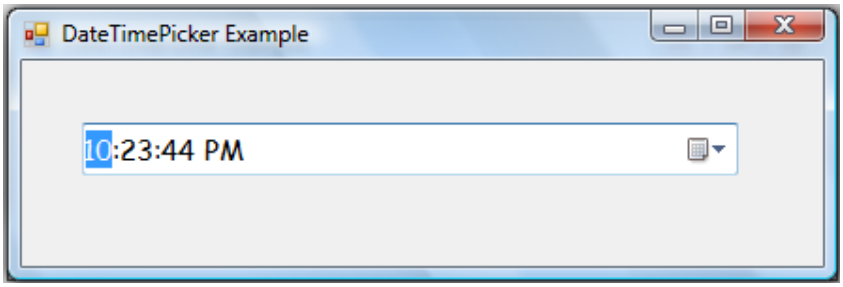
Example 2:

If the format property is changed to short, the value in the control will look as follows:



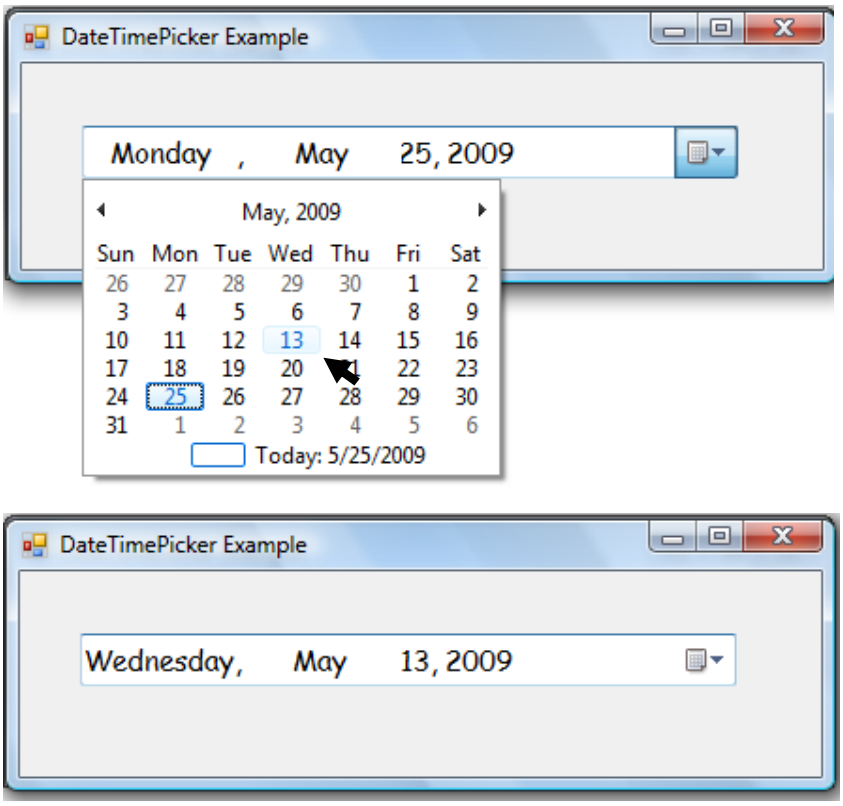
Example 3:

If the format property is changed to time, the value in the control will look as follows:



The value displayed in the tool will be that of the current date and time, but it is possible that the user can select another value by clicking on the control's list arrow button as can be seen in the next example:

Example 4:



New values can also be selected by using the arrow keys on the keyboard. The left and write arrow keys will toggle between the month, day and year. E.g. the day can be selected. The up and down arrow keys will now select either the previous value or next value.

E.g. Select 13. By pressing the up arrow, this value will be changed to 14 etc.

The value selected can be determined by the DateTimePicker control's Value property. This value will be in the following format:

MM/DD/YYYY HH:MM AM/PM

E.g. 5/4/2009 9:32 PM
5/25/2009 10:34 PM etc.

5.9 DATETIMEPICKER EXAMPLE

Create a user interface with 2 DateTimePicker controls. Ask the user to select a date in the first control as well as a date in the second control. The first date will indicate the day on which guests will arrive at ENJOY GUEST HOUSE and the second date will be the date that they will depart. For the sake of simplicity, let's assume that both dates must be in the same month. Calculate the number of days that the guests will stay and the total cost for the room. A double room is R123.50 per night and a single room 102.50 per night.

Code in the code editor:

```
Public Class DateTime

Private Sub DateTime_Load(. . .) Handles MyBase.Load
   'Select the single room radio button as the default radio button
   'when the form is loaded
   radSingle.Checked = True
End Sub

Private Sub btnCalc_Click(. . .) Handles btnCalc.Click
   Dim decCost, decAmtDue As Decimal
   Dim intNoDays As Integer

   'Declare two string variables to contain the 2 dates selected
   Dim strDateArrive, strDateDepart As String

   'Declare 2 variables to contain the 2 days selected
   Dim intDayArrive, intDayDepart As Integer

   'Declare 2 variables to contain the 2 months selected
   Dim intMonthArrive, intMonthDepart As Integer

   'Place the 2 selected days in the two string variables
   strDateArrive = dtpDateArrive.Value
   strDateDepart = dtpDateDepart.Value

   'Use the substring method to obtain the days and month values separately
   Dim intPosition As Integer
   intPosition = strDateArrive.IndexOf("/")
   intMonthArrive = _
               Convert.ToInt32(strDateArrive.Substring(0, intPosition))
   strDateArrive = strDateArrive.Remove(0, intPosition + 1)
   intPosition = strDateArrive.IndexOf("/")
   intDayArrive = Convert.ToInt32(strDateArrive.Substring(0, intPosition))
   intPosition = strDateDepart.IndexOf("/")
   intMonthDepart = _
               Convert.ToInt32(strDateDepart.Substring(0, intPosition))
   strDateDepart = strDateDepart.Remove(0, intPosition + 1)
   intPosition = strDateDepart.IndexOf("/")
   intDayDepart = Convert.ToInt32(strDateDepart.Substring(0, intPosition))

   'Determine the cost per night
   If radSingle.Checked Then
      decCost = 102.5D
   Else
      decCost = 123.5D
   End If

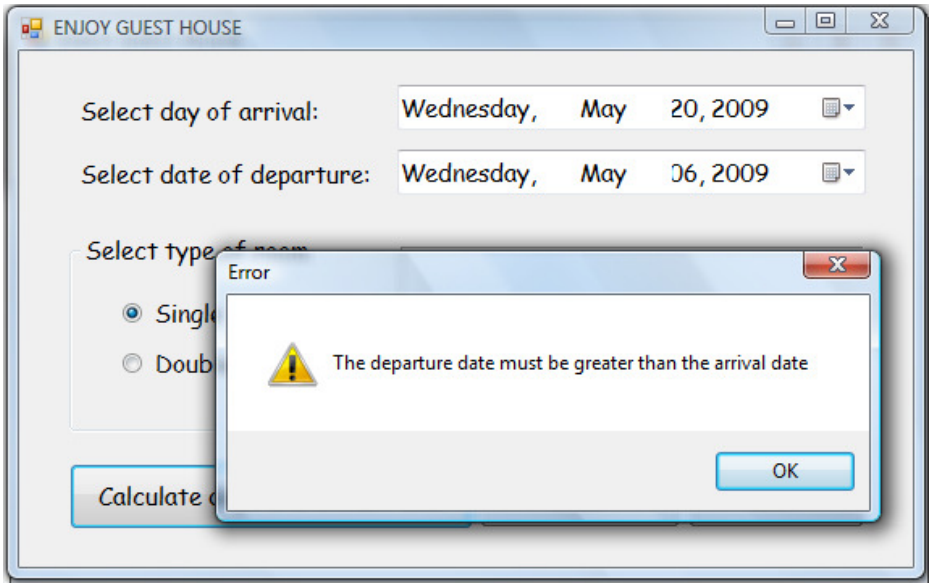
   'Test that months must be the same for the sake of this example
   'Date processings could become quite complex to cater for all
   'possibilities
   If intMonthArrive = intMonthDepart Then
      If intDayDepart > intDayArrive Then
         intNoDays = intDayDepart - intDayArrive
         decAmtDue = intNoDays * decCost
         lblAnswer.Text = "Number of days: " & intNoDays & _
                          ControlChars.NewLine & _
                         "Amount Due = R" & decAmtDue.ToString("N2")
      Else
```

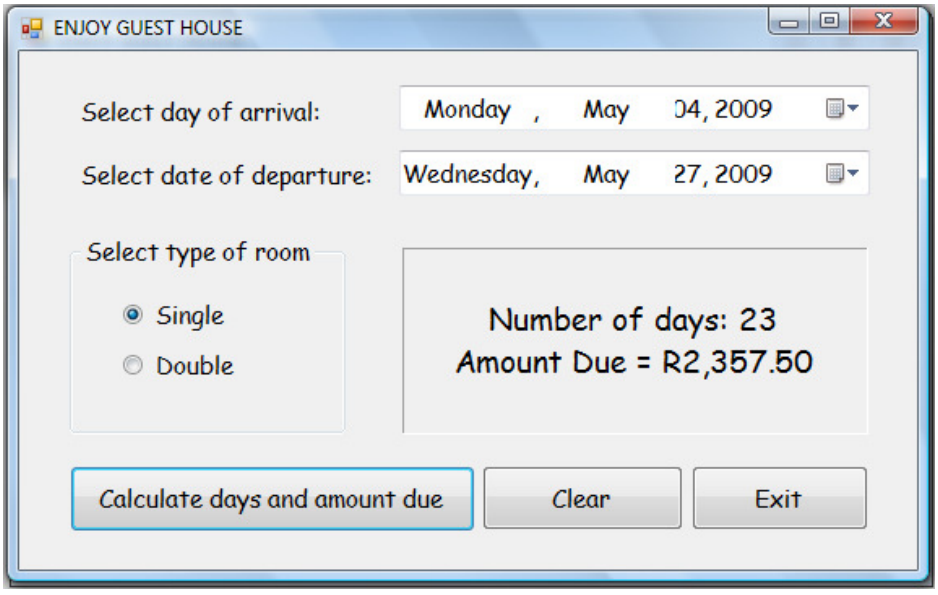
```
            MessageBox.Show("The departure date must be greater than the
                  arrival date", "Error", MessageBoxButtons.OK, _
                  MessageBoxIcon.Exclamation)
         End If
      Else
         MessageBox.Show("The two months must be the same", "Error", _
                     MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
      End If
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   lblAnswer.Text = ""
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Possible output:





CHAPTER 6
Comprehensive Examples

6.1 INTRODUCTION

We have studied different topics in the level 2 and now, in the level 3 courses. In this chapter we are going to combine different topics and plan and write more comprehensive Visual Basic.NET programs.

OUTCOMES

When you have studied this chapter you should be able to:

- Combine various techniques and statements to solve problems of a more complex nature.

6.2 Examples of comprehensive programs

Plan and write Visual Basic.NET programs to solve the following problems:

6.2.1 EXAMPLE 1

This program includes:

- A main procedure
- Sub procedures
- Value and reference parameters
- Arrays
- Selection structure
- Other VB.NET statements

The 250 students of the GOOD RESULTS COLLEGE have to select the chairman and the vice chairman of the new student representative council. Six candidates have been nominated for these positions. All the registered students may partake in the selection and a student has to indicate for which candidate (1 to 6) he/she wants to vote. The candidate who has the most votes will be the chairman and the vice chairman will have the 2nd most votes.

The main procedure contains 2 parallel arrays. The first array contains the 6 names of the candidates en the second array will contain the number of votes cast by the students for a specific candidate.

The votes of the students are entered in a sub procedure. A student votes for his/her number 1 candidate but also for the number 2 candidate.

The program adds two votes for a chairman vote to the correct position in the votes array and one vote for a vice-chairman vote to the correct position in the votes array (this is done in main).

A loop will continue to enter the votes of the students until all the students have cast their votes.

The chairman and vice chairman's names will be selected in the main program and then displayed in a sub procedure before the program terminates.

For the sake of entering data into the program we will test this program with only 10 students.

Planning:

	Description	Type	Name of variable
Sub Procedure (InputVotes):			
Parameters:			
	Address of chairman vote	integer	refChair
	Address of vice-chairman vote	integer	refViceChair
Sub Procedure (DisplayResult):			
Parameters:			
	Copy of chairman name	string	valChairman
	Copy of vice chairman name	string	valViceChair
Output:	name of chairman	string	valChairman
	name of vice-chairman	string	valViceChair
Main procedure (VoteForChairman):			
Has 2 arrays:	Array for names	string	nameArray
	Array for votes	integer	votesArray
Input:	vote for chairman	integer	MainChair
	vote for vice-chairman	integer	MainVice
Output:	name of chairman	string	Chair
	name of vice-chairman	string	Vice

Algorithm:

```
SubProcedure InputVotes(refChair, refVice)                    ~ input votes
        display "Enter the number of the person you choose for chairman – 1 to 6"
        enter refChair
        display ""Enter the number of the person you choose for vice chairman – 1 to 6"
        enter refViceChair
end Sub
```

```
Sub Procedure DisplayResult (valChairman, valViceChairman)
        display "The chosen chairman is ", valChairman
        display "The chosen vice-chairman is  ", valViceChair
end Sub

Main procedure  VoteForChairman

        ~       The array nameArray contains the names of the 6 members
        ~       The array votesArray contains a zero in every element

        for k = 1 to 10
                call InputVotes(MainChair, MainVice)
                ~       add to votes in array
                ~       add 2 for chairman and 1 for vice-chairman
                votesArray(MainChair-1) = votesArray(MainChair-1) + 2
                votesArray(MainVice-1) = votesArray(MainVice-1) + 1
        next

        ~       determine chairman
        Most = votesArray(0)
        Chairman = nameArray(0)
        position = 0
        for i = 1 to 5                          ~  we have already used the first element
                if votesArray(i) > Most
                        Most = votesArray(i)
                        Chairman = nameArray(i)
                        Position = i
                endif
        next

        ~       determine vice-chairman
        votesArray(position) = 0                ~ eliminate chairman
        Most = votesArray(0)
        Vice = nameArray(0)
        for i = 1 to 5                          ~  we have already used the first element
                if votesArray(i) > Most
                        Most = votesArray(i)
                        Vice = nameArray(i)
                endif
        next

        DisplayResult ( Chairman, Vice)
End
```

VB.NET Solution

Code in the code editor:

```
Public Class VoteForChairman

Private strNames() As String = {"John Smythe", "Derick Butley", _
                                "Mpho Hlonghlane", "Shaun Dry", _
                                "Axel Moekoena", "Tshepo Maake"}
Private intVotes() As String = {0, 0, 0, 0, 0, 0}

Private Sub InputVotes(ByRef intChair As Integer, ByRef intVice As Integer)
   Dim blnValidChair, blnValidVice As Boolean
   Dim strChair, strVice As String
   Do
     strChair = InputBox("Enter the number of the person you choose for
                          chairman", "Enter a number 1 to 6")
     intChair = Convert.ToInt32(strChair)
     If intChair >= 1 AndAlso intChair <= 6 Then
        blnValidChair = True
     Else
        blnValidChair = False
        MessageBox.Show("Enter only a number from 1 to 6", _
                        "Please re-enter", MessageBoxButtons.OK, _
                        MessageBoxIcon.Exclamation)
     End If
   Loop Until blnValidChair
   Do
     strVice = InputBox("Enter the number of the person you choose for vice
                         chairman", "Enter a number 1 to 6")
     intVice = Convert.ToInt32(strChair)
     If intVice >= 1 AndAlso intVice <= 6 Then
        blnValidVice = True
     Else
        blnValidVice = False
        MessageBox.Show("Enter only a number from 1 to 6", _
                        "Please re-enter", MessageBoxButtons.OK, _
                        MessageBoxIcon.Exclamation)
     End If
   Loop Until blnValidVice
End Sub

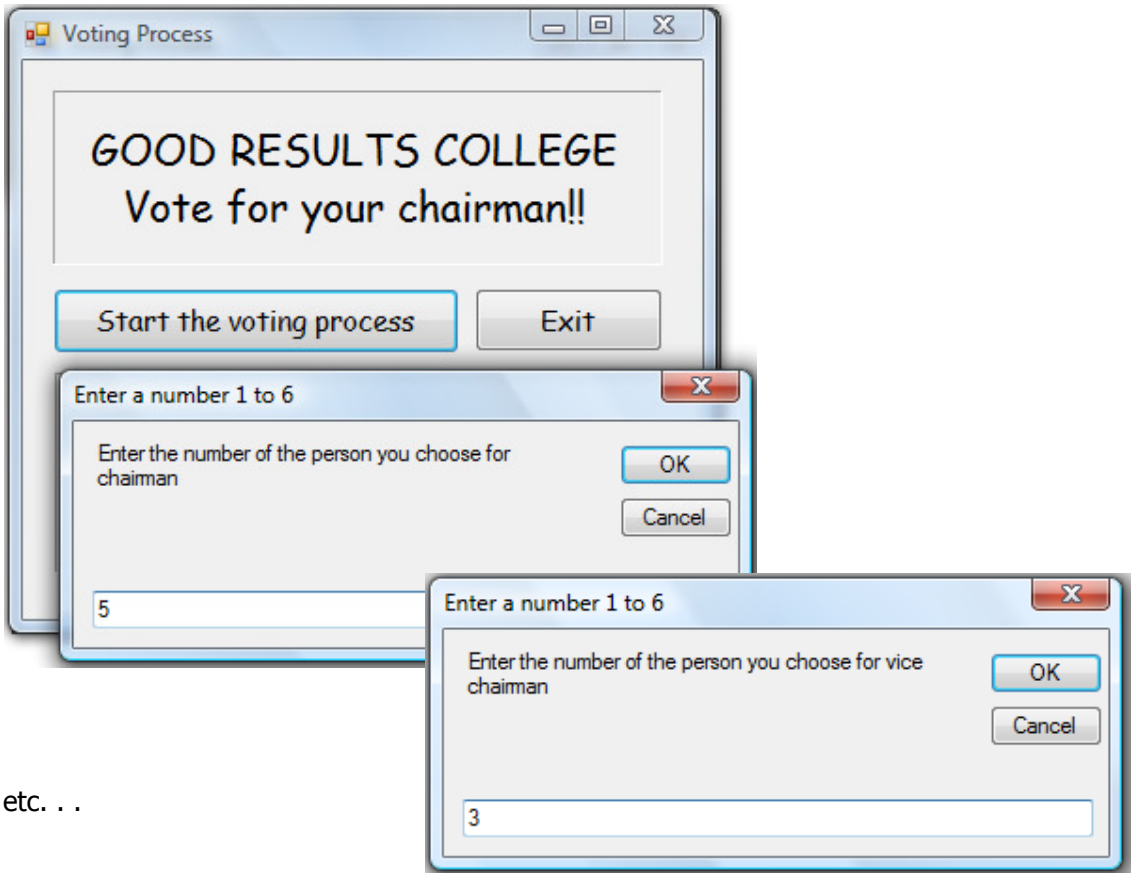
Private Sub DisplayResult(ByVal strChairman As String, _
                          ByVal strViceChairman As String)
  lblOutcome.Text = "Congratulations!!" & ControlChars.NewLine & _
                    "The chosen chairman is " & _
                    strChairman & ControlChars.NewLine & _
                    "The chosen vice-chairman is " & strViceChairman
End Sub

Private Sub btnStart_Click(. . .) Handles btnStart.Click
   Dim intMainChair, intMainVice As Integer
   Dim intMost, intPosition As Integer
   Dim strChairman, strVice As String
   For k As Integer = 1 To 10
       Call InputVotes(intMainChair, intMainVice)
       intVotes(intMainChair - 1) += 2
       intVotes(intMainVice - 1) += 2
   Next
   intMost = intVotes(0)
```

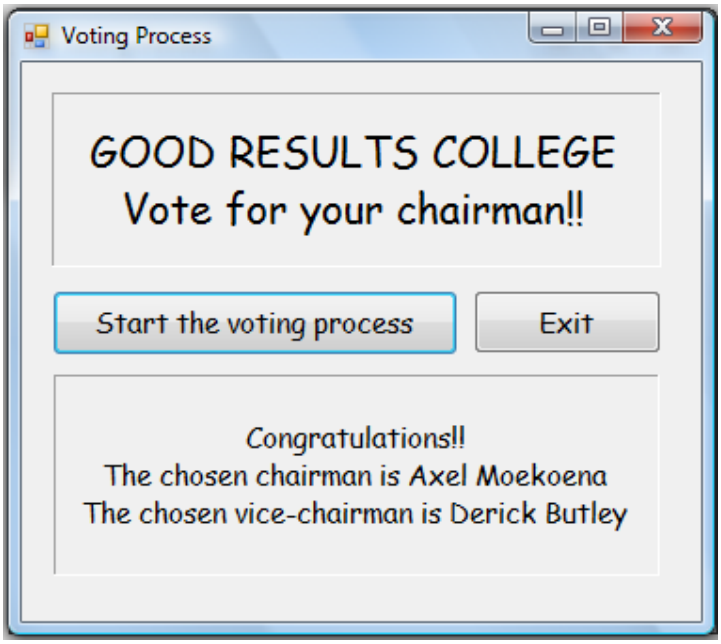
```
   strChairman = strNames(0)
   intPosition = 0
   For i As Integer = 1 To 5
       If intVotes(i) > intMost Then
          intMost = intVotes(i)
          strChairman = strNames(i)
          intPosition = i
       End If
   Next
   intVotes(intPosition) = 0
   intMost = intVotes(0)
   strVice = strNames(0)
   For i As Integer = 1 To 5
       If intVotes(i) > intMost Then
          intMost = intVotes(i)
          strVice = strNames(i)
       End If
   Next
   Call DisplayResult(strChairman, strVice)
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Examples of output:



etc. . .



6.2.2 EXAMPLE 2

This program includes:

- A main procedure
- Two functions
- Value parameters
- Nested if statement
- Nested loops
- Calculations

Sakkie wants to collect widgets. He may only spend 7.5% of his net monthly salary to buy these widgets. Plan and write a VB.NET program to calculate how many widgets he can afford to buy in 24 months. His gross annual salary must be entered and his deductions must be subtracted before he may calculate how much he may spend on the widgets. The user also has to enter the price of one widget.

Tax is payable according to the following table:

Monthly Salary	Percentage Tax
0 – 4999.99	0
5000.00 – 10000	5.5%
10000.01 – 20000.00	15%
20000.01 and more	27%

He also has the following deductions:

Medical aid	monthly salary less than 10000	R250
	monthly salary 10000.01 and more	R500
Union contribution	R60	
UIF	1% of gross monthly salary	

Use a function to calculate the tax and another function to calculate the sum of all the deductions.

You may assume that he will earn the same salary for the first 12 months, and then getting an increase of 6%. Display the number of widgets he will be able to buy.

Planning:

	Description	Type	Name of variable
Function (CalculateTax)			
Input:	copy of monthly salary	real	valMonSalary
Output:	amount of tax	real	taxAmt
Function (CalculateTotalDeduction)			
Input:	copy of monthly salary	real	valMonSalary
	copy of tax amount	real	valTax
Output:	total deductions	real	totDeduct
Main procedure (CalculateNoWidgets)			
Input:	annual salary	real	anSalary
	price of one widget	real	widPrice
Output:	number of widgets	integer	widgets

Algorithm:

```
Function CalculateTax (valMonSalary)          ~     calculate the tax
        if valMonSalary < 5000
                taxAmt = 0
        else
                if valMonSalary < 10000.01
                        taxAmt = valMonSalary * 0.055
                else
                        if valMonSalary < 20000.01
                                taxAmt = valMonSalary * 0.15
                        else
                                taxAmt = valMonSalary * 0.27
                        endif
                endif
        endif
return taxAmt
```

```
Function CalculateTotalDeduction (valMonSalary, valTax)
        if valMonSalary < 10000
                medical = 250
        else
                medical = 500
        endif
        totDeduct = valTax + medical + 60 + (valMonSalary * 0.01)
return totDeduct

CalculateNoWidgets          ~       main program
        sum = 0             ~       initialize amount to buy widgets
        display "Provide the annual salary:"
        enter anSalary
        display "Provide the price of one widget:"
        enter widPrice

        monthSal = anSalary / 12

        for yr = 1 to 2         ~       repeat for 2 years
                ~       repeat  for first 12 months
                for mon = 1 to 12
                        tax = CalculateTax(monthSal)        ~  call function to calculate tax
                        deductions = CalculateTotalDeductions(monthSal, tax)
                                                    ~  call function to calc total deduct
                        netSal = monthSal – deductions

                        sum = sum + netSal * 0.075          ~  add to total
                next
                monthSal = monthSal * 1.06          ~       increase salary by 6%
        next

        widgets = sum \ widPrice                    ~       do integer division
        display "Sakkie can buy ", widgets, " widgets"
end
```

VB.NET SOLUTION

Code in the code editor:

```
Public Class CalcNoWidgets

Private Function CalculateTotalDeductions(ByVal decMonSalary As Decimal,_
                 ByVal decTax As Decimal) As Decimal
   Dim decMedical As Decimal
   Dim decTotalDeductions As Decimal
   If decMonSalary < 10000D Then
      decMedical = 250D
   Else
      decMedical = 500D
   End If
   decTotalDeductions = decTax + decMedical + 60D + (decMonSalary * 0.01)
   Return decTotalDeductions
End Function
```

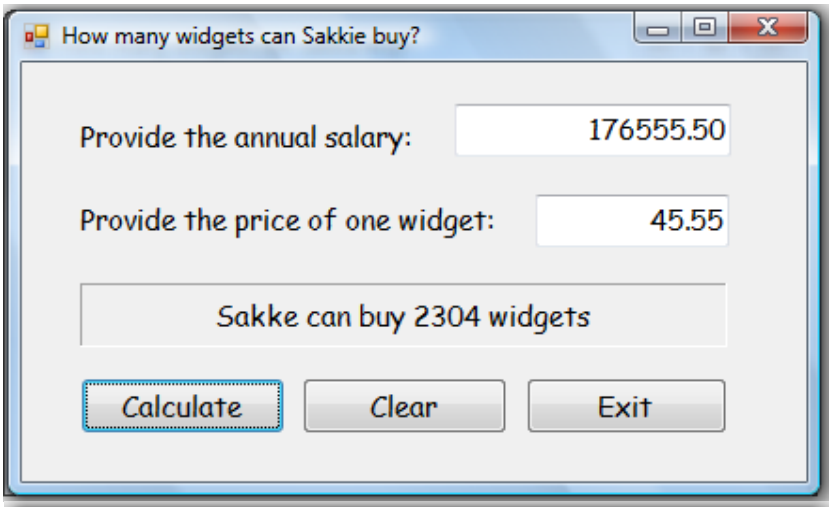
```
Private Function CalculateTax(ByVal decMonSalary As Decimal) As Decimal
   Dim decTaxAmt As Decimal
   If decMonSalary < 5000D Then
      decTaxAmt = 0
   Else
      If decMonSalary < 10000.01D Then
         decTaxAmt = decMonSalary * 0.055
      Else
         If decMonSalary < 20000.01D Then
            decTaxAmt = decMonSalary * 0.15D
         Else
            decTaxAmt = decMonSalary * 0.27
         End If
      End If
   End If
   Return decTaxAmt
End Function

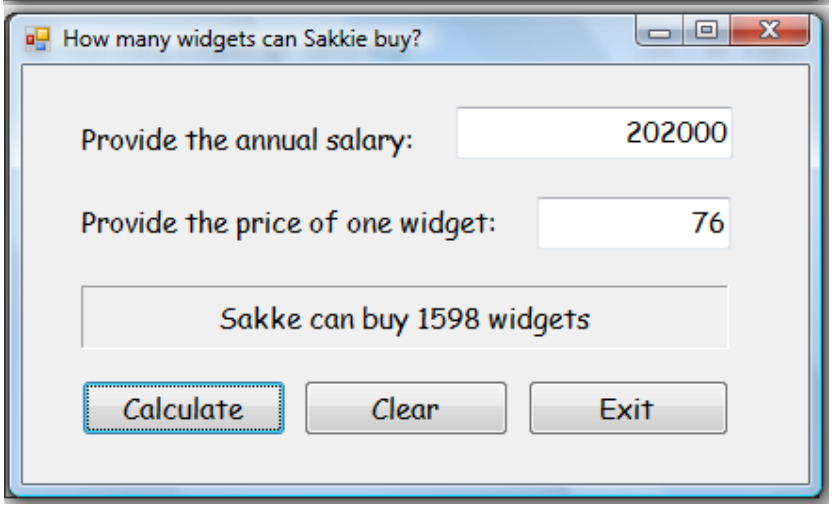
Private Sub btnCalc_Click(. . .) Handles btnCalc.Click
   Dim decSum As Decimal = 0
   Dim decAnSalary, decWidPrice As Decimal
   Dim decMonthSal, decNetSal, decTax, decDeductions As Decimal
   Dim intWidgets As Integer
   'Enter the annual salary and price per widget.
   'Assume values will be valid.
   decAnSalary = Convert.ToDecimal(txtAnSal.Text)
   decWidPrice = Convert.ToDecimal(txtWidPrice.Text)
   decMonthSal = decAnSalary / 12
   For yr As Integer = 1 To 2
       For mon As Integer = 1 To 12
           decTax = CalculateTax(decMonthSal)
           decDeductions = CalculateTotalDeductions(decMonthSal, decTax)
           decNetSal = decMonthSal - decDeductions
           decSum = decSum + decNetSal * 0.075
       Next
       decMonthSal = decMonthSal * 10.06
   Next
   intWidgets = decSum \ decWidPrice
   lblAnswer.Text = "Sakke can buy " & intWidgets.ToString & " widgets"
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   txtAnSal.Text = ""
   txtWidPrice.Text = ""
   lblAnswer.Text = ""
   txtAnSal.Focus()
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Examples of output:





6.2.3 EXAMPLE 3

This program includes:

- A main procedure
- Two-dimensional array
- Parallel arrays
- Value parameters
- Loop
- Function
- Sub procedure
- Validating
- Select statement

Do the planning and then write a VB.NET program to solve the following problem:
The fishermen club has a competition where 6 fishermen competed to determine who the champion fisherman is. The fish are categorized as small, medium or large.

The program must input the number of the fisherman who caught a fish and as well as the size (S, M or L) of the fish. Ensure that the fisherman code is correct and that a correct code is entered for the fish.

Every time a fish is caught, a 2 dimensional array called pointsArray is updated. This array must not be declared as global, but in the main procedure. You may assume that at least one fish will be caught.

A parallel array called nameArray that is declared in the main procedure, contains the names of the 6 fishermen.

Diagram of 2 dimensional array:

Size / Fisherman	Small	Medium	Large
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0

If fisherman 4 caught a medium fish and fisherman 2 caught a small fish, the updated table must contain the following values:

Size / Fisherman	Small	Medium	Large
1	0	0	0
2	1	0	0
3	0	0	0
4	0	1	0
5	0	0	0
6	0	0	0

To determine the winner the sum of the points are calculated for every fisherman as follows: Pass the 3 values for every fisherman to a function called CalcPoints where 1 point is assigned for a small fish, 1.5 points for a medium fish and 1.9 points for a large fish. Return the sum. Determine the winner and his name, in the main procedure.

It is not known how many fish will be caught on the competition day and the program terminates when fisherman code = 9 is entered.

The sub procedure DisplayResult will display the name of the winner and the number of points collected.

Planning:

	Description	Type	Name of variable
Function (CalcPoints)			
Input:	copy of no of small fish	integer	valSmall
	copy of no of medium fish	integer	valMedium
	copy of no of large fish	integer	valLarge
Output:	points earned by fisherman	real	points
Sub procedure (DisplayResult)			
Input:	copy of name of winner	string	valName
	copy of total points	real	valPoints
Output:	name of winner and points	string	- none -
FishermanCompetition	~	main procedure	
Input:	number of fisherman	integer	noFishMan
	size of fish	character	size
Output:	error message	string	- none -

Algorithm:

```
Function CalcPoints (valSmall, valMedium, valLarge)

    points = valSmall + valMedium * 1.5 + valLarge * 1.9      ~  convert to real

Return points

Sub Procedure DisplayResult (valName, valPoints)

    display "And the winner is . . . "
    display valName, " with ", valPoints, " points"

End sub

FishermanCompetition                                          ~ main procedure

    Declare 2 dim array pointsArray to store the points per fisherman
    Declare a paired array to store the names of the fisherman

    display "Enter the number of the fisherman that caught the first fish, 9 to terminate"
    enter noFishMan
```

```
    do while (noFishMan <> 9)
        if noFishMan >= 1 And noFishMan <= 6
            display "Enter the size of the fish, S, M or L"
            enter size
            do while (size <> "S" OR size <> "M" OR size <> "L")
                display "You have entered an incorrect code, try again"
                enter size
            loop

            select case size
                case "S"
                    cIndex = 0
                case "M"
                    cIndex = 1
                case "L"
                    cIndex = 2
            endselect

            pointsArray(noFishMan-1, cIndex) =
                    pointsArray(noFishMan-1, cIndex) + 1
            ~       increase corresponding element by 1
        Else
            Display "Please enter a number from 1 to 6"
        Endif
        display "Enter the number of the fisherman that caught the next fish"
        display "Enter 9 to terminate"
        enter noFishMan
    loop
    winner = 0
    for w = 0 to 5
        points = CalcPoints(pointsArray(w,0), pointsArray(w,1), pointsArray(w,2))
        if points > winner          ~  will be at least once
            winner = points
            winnerName = nameArray(w)
        endif
    next

    call DisplayResult(winnerName, winner)
end
```

VB.NET SOLUTION:

Code in the code editor:

```
Public Class FishermanCompetition

Private Function CalcPoints(ByVal intSmall As Integer, _
        ByVal intMedium As Integer, ByVal intLarge As Integer) As Double
   Dim dblPoints As Double
   dblPoints = Convert.ToDouble(intSmall + intMedium * 1.5 + _
                               intLarge * 1.9)
   Return dblPoints
End Function
```

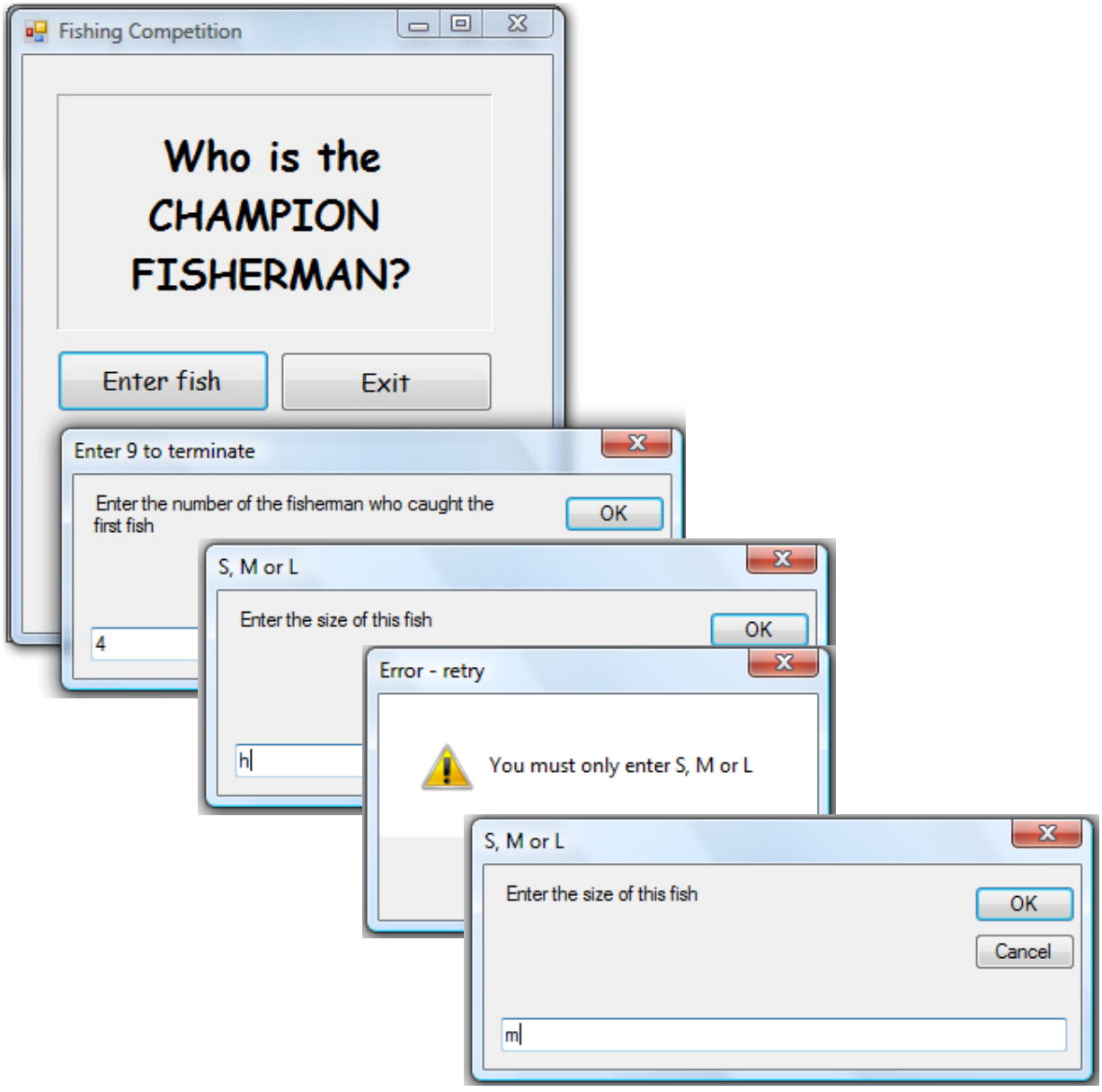
```
Private Sub DisplayResults(ByVal StrName As String, _
                           ByVal dblPoints As Double)
   blChampion.Text = "And the winner is . . . " & ControlChars.NewLine & _
             StrName & " with " & dblPoints.ToString("N1") & " points"
End Sub

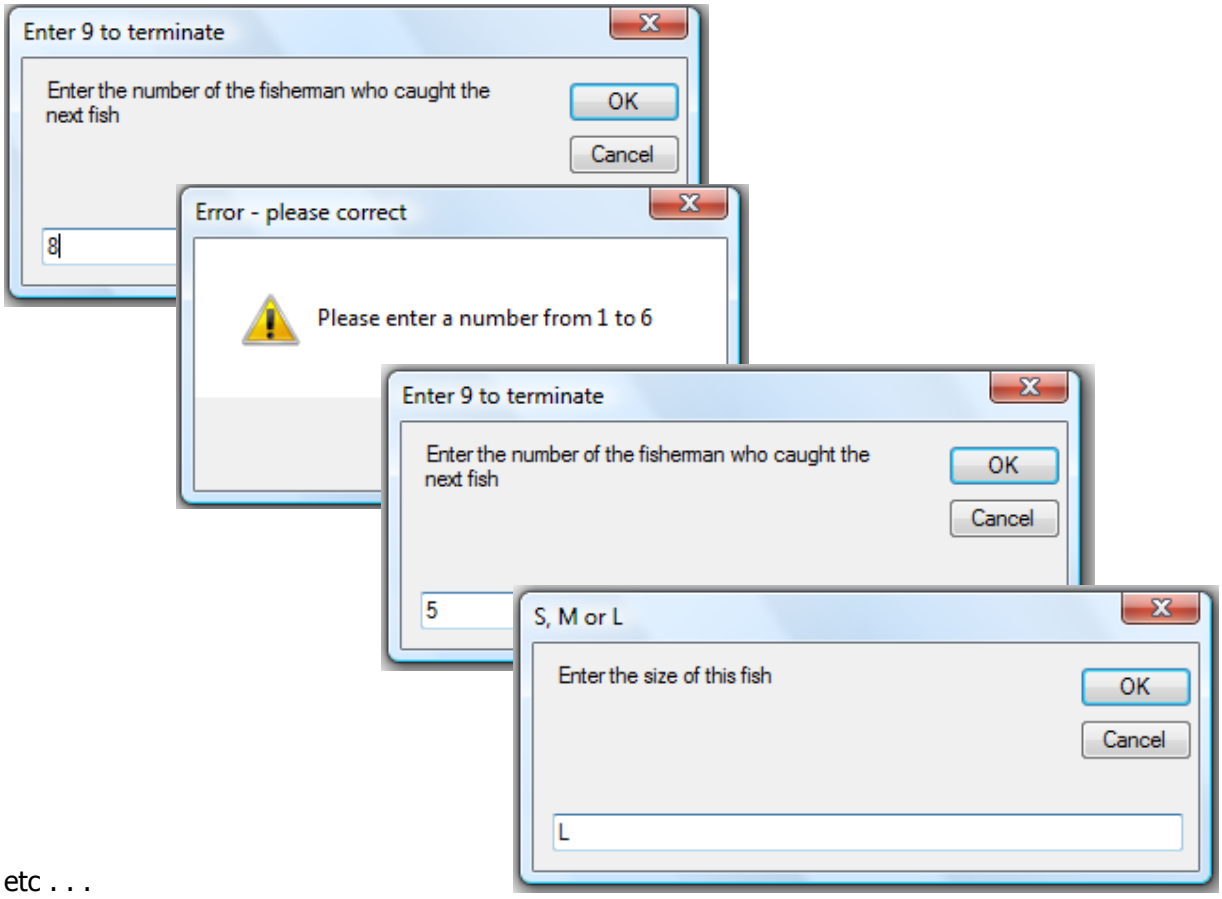
Private Sub btnEnter_Click(. . .) Handles btnEnter.Click
   Dim intPointsArray(,) = {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}, _
                            {0, 0, 0}, {0, 0, 0}, {0, 0, 0}}
   Dim strNameArray() = {"Big Ben", "Jolly James", "Fast Fred", _
                         "Eager Edward", "Tiger Terry", "Rough Rufus"}
   Dim intNoFishMan As Integer
   Dim strSize As String
   Dim strWinnerName As String
   Dim dblWinner, dblPoints As Double
   Dim blnValidSize As Boolean
   Dim strFishMan As String
   Dim intCIndex As Integer
   strFishMan = InputBox("Enter the number of the fisherman who caught the
                         first fish", "Enter 9 to terminate")
   Do While strFishMan <> "9"
      intNoFishMan = Convert.ToInt32(strFishMan)
      If intNoFishMan >= 1 AndAlso intNoFishMan <= 6 Then
      Do
         strSize = InputBox("Enter the size of this fish", "S, M or L")
         If strSize.ToUpper = "S" Or strSize.ToUpper = "M" Or _
            strSize.ToUpper = "L" Then
            blnValidSize = True
         Else
            MessageBox.Show("You must only enter S, M or L", _
                            "Error - retry", MessageBoxButtons.OK,_
                            MessageBoxIcon.Exclamation)
            blnValidSize = False
         End If
      Loop Until blnValidSize
      Select Case strSize
         Case "S", "s"
           intCIndex = 0
         Case "M", "m"
           intCIndex = 1
         Case "L", "l"
           intCIndex = 2
      End Select
      intPointsArray(intNoFishMan - 1, intCIndex) = _
             intPointsArray(intNoFishMan - 1, intCIndex) + 1
   Else
      MessageBox.Show("Please enter a number from 1 to 6", _
                "Error - please correct", MessageBoxButtons.OK, _
                MessageBoxIcon.Exclamation)
   End If
   strFishMan = InputBox("Enter the number of the fisherman who caught the
                         next fish", "Enter 9 to terminate")
  Loop
```

```
  dblWinner = 0
  For w As Integer = 0 To 5
    dblPoints = CalcPoints(intPointsArray(w, 0), intPointsArray(w, 1), _
                           intPointsArray(w, 2))
    If dblPoints > dblWinner Then
       dblWinner = dblPoints
       strWinnerName = strNameArray(w)
    End If
  Next
  Call DisplayResults(strWinnerName, dblWinner
End Sub

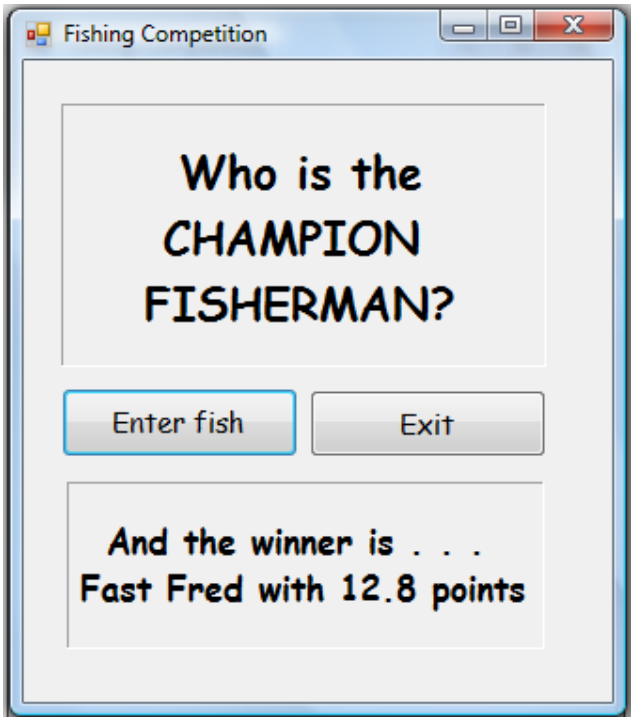
Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Examples of output:




Enter 9 to terminate
Enter the number of the fisherman who caught the next fish
OK
Cancel
8
Error - please correct
Please enter a number from 1 to 6
Enter 9 to terminate
Enter the number of the fisherman who caught the next fish
OK
Cancel
5
S, M or L
Enter the size of this fish
OK
Cancel
L

etc . . .


Fishing Competition
Who is the CHAMPION FISHERMAN?
Enter fish
Exit
And the winner is . . .
Fast Fred with 12.8 points

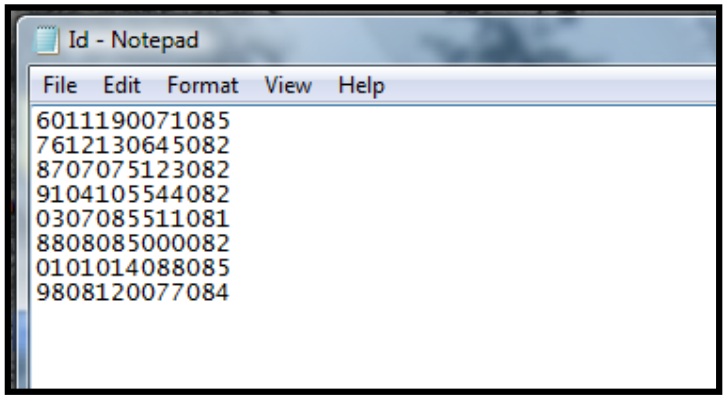
6.2.4 EXAMPLE 4

This program includes:

- A main procedure
- A Sub procedure
- Value and reference parameters
- Reading text lines from a sequential access file
- Loop
- String Manipulation
- Date processing with DateTimePicker

Some id-numbers are stored on the file called "C:Id.Txt". Read these id-numbers from the file and display every id number in a list box, followed by the birth date of this person in the format DD Month YYYY as well as the age of this person in years and months. Make use of a DateTimePicker to determine today's date.

Example of text file:



Code in the code editor:

```
Public Class FindBirthday

Private Sub CalcAge(ByVal intTYYYY As Integer, ByVal intTMM As Integer, _
                    ByVal intBYYYY As Integer, ByVal intBMM As Integer, _
                    ByRef intAgeYY As Integer, ByRef intAgeMM As Integer)
   If intBMM > intTMM Then
      intTYYYY = intTYYYY - 1
      intTMM = intTMM + 12
   End If
   intAgeYY = Math.Abs(intBYYYY - intTYYYY)
   intAgeMM = Math.Abs(intBMM - intTMM)
End Sub

Private Sub btnEnter_Click(. . .) Handles btnEnter.Click

   Dim strIdNumber As String

   'Declare variables for age
   Dim intAgeYY, intAgeMM As Integer
```

```
'Birthdate termined from id number and separated into DD, MM and YYYY
Dim strBirthDate As String
Dim intPosition As Integer
Dim strDD, strYYYY, strYY As String
Dim chaD As Char
Dim intMM, intYY, intYYYY As Integer

'today's date from dateTimePicker
Dim strToday, strtodayYYYY, strTodayYY As String
Dim intTodayMM, intTodayYY, intTodayYYYY As Integer

'Declaration of array with months
Dim strMonth() As String = {"January", "February", "March", "April", _
             "May", "June", "July", "August", "September", "October", _
             "November", "December"}

'Get today's date and divide into DD, MM and YYYY
strToday = dtpToday.Value
intPosition = strToday.IndexOf("/")
intTodayMM = Convert.ToInt32(strToday.Substring(0, intPosition))
strToday = strToday.Remove(0, intPosition + 1)
intPosition = strToday.IndexOf("/")
strToday = strToday.Remove(0, intPosition + 1)
strtodayYYYY = strToday.Substring(0, 4)
intTodayYYYY = Convert.ToInt32(strtodayYYYY)
strTodayYY = strtodayYYYY.Substring(2, 2)
intTodayYY = Convert.ToInt32(strTodayYY)

'Declare StreamReader object
Dim IDFile As System.IO.StreamReader

'Test if the file exist, do processing only if it exists
If System.IO.File.Exists("C:\Id.txt") Then

    'open file to read id's from it
    IDFile = System.IO.File.OpenText("C:\ID.txt")

    'display Headings
    lstDisplay.Items.Add("ID NUMBER".PadRight(15) & _
                         "BIRTH DATE".PadRight(18) & "AGE")
    lstDisplay.Items.Add(" ")

    'Read first id number from file
    strIdNumber = IDFile.ReadLine()

    'Loop for all id's
    Do While IDFile.Peek <> -1

       'Use string manipulation to get YY, MM and DD from id number
        strYY = strIdNumber.Substring(0, 2)
        intYY = Convert.ToInt32(strIdNumber.Substring(0, 2))
        intMM = Convert.ToInt32(strIdNumber.Substring(2, 2))
        strDD = strIdNumber.Substring(4, 2)

        'Add century for display purposes
        If intYY > intTodayYY Then
           strYYYY = "19" & strYY
        Else
           strYYYY = "20" & strYY
        End If
        intYYYY = Convert.ToInt32(strYYYY)
```

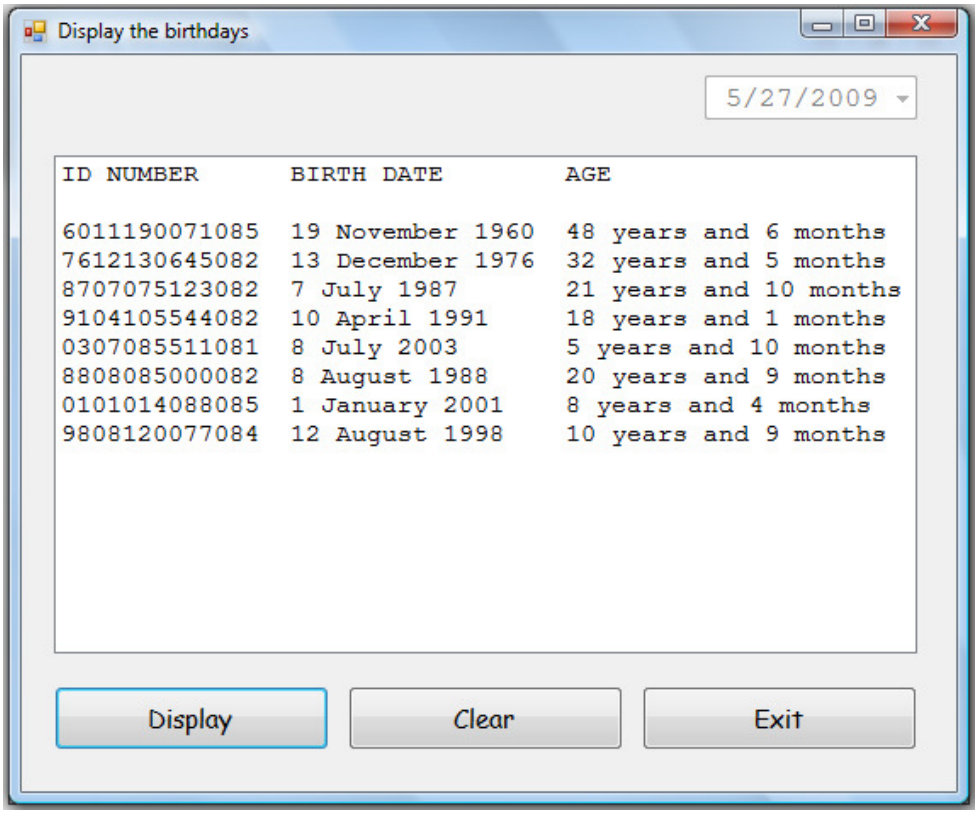
```
            'Prepare birht date for display purposes
            'Remove leading zero from day if necessary
            strBirthDate = strMonth(intMM - 1) & " " & strYYYY
            If strDD.Substring(0, 1) = "0" Then
               chaD = strDD.Substring(1, 1)
               strBirthDate = strBirthDate.Insert(0, chaD & " ")
            Else
               strBirthDate = strBirthDate.Insert(0, strDD & " ")
            End If

            'Call a sub procedure to determine age
            Call CalcAge(intTodayYYYY, intTodayMM, intYYYY, intMM, _
                         intAgeYY, intAgeMM)
            lstDisplay.Items.Add(strIdNumber.PadRight(15) & _
                 strBirthDate.PadRight(18) & intAgeYY.ToString & _
                 " years and " & intAgeMM.ToString & " months")
            strIdNumber = IDFile.ReadLine()
         Loop
         IDFile.Close()
      Else
         MessageBox.Show("File with member names does not exist", _
              "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
      End If
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   lstDisplay.Items.Clear()
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Output:





Practical

1. Rewrite the program 6.2.1 to determine the chairman and vice-chairman by sorting the arrays. Remember that you have to sort the names as well! You may use the bubble sort method or the selection sort method.

2. Have another look at the program 6.2.1. It is possible that a student may vote for the same person for chairman as well as for vice-chairman. In a case like this 3 points will be added to the total instead of just one or two which is incorrect. Include another function to test if the student voted for the same person. If so, display a suitable error message and ask the student to re-enter the votes until the student has voted for two different people.

3. Plan a solution to solve this problem and then write the VB.NET program for the following:

   Five salesmen sell goods on a commission basis for the company EVERYBODY'S DELIGHT during February. They earn commission on their daily sales according to the following table:

Sales in R/c	**Commission**
0 – 100.00	2%
100.01 – 24999.99	15%
25000.00 and more	32%

   The input to this program is the number of the salesman and his sales of the day. The program must calculate the daily commission and add it to the commission of the month in the corresponding element of a one dimensional array e.g.

   SalesArray for commission of 5 salesmen at the beginning of the month

0.00	0.00	0.00	0.00	0.00

   The number of transactions per day is unknown and there may be transactions for every day of the month. The programmer has to decide how to indicate the end of the transactions.

   The salesmen decided that they are only going to work from February to December because nobody has money to buy anything in January. So, they take leave in January. To provide for an income in January, they decided to save a fixed amount that can be invested at a rate of 8% per annum, during the other months. Every salesman wants to have R10000.00 for January (use the PMT method).

The program must display:

- how much money every salesman will have at the end of this month after the saving is deducted
- how much money must be saved for January

Use functions and sub procedures where possible and program effectively.

4. Declare a 2 dimensional integer array with 7 rows and 10 columns. Write code to fill the 70 elements with random integers between 10 and 90. Write also code to calculate and display
   - the average of every row
   - the average of every column
   - the average of all the elements

   The answers must be displayed in a manner that is easily understood.

CHAPTER 7

Error Handling and Debugging Techniques

7.1 INTRODUCTION

It is not expected of the beginner programmer to write flawless programs! The best way to learn to program without errors is to practice, practice and once more practice. It is said that practice makes perfect!

There are however, certain techniques to assist a programmer to find errors in program.

OUTCOMES

When you have studied this chapter you should be able to:

- distinguish between different types of errors
- use a trace table to test the correctness of a program before the program is coded in VB.NET
- display the value of specific variables where errors in calculations occurred
- how to use stubs when developing large applications

7.2 DIFFERENT TYPES OF ERRORS

7.2.1 SYNTAX ERRORS

At the moment you are reading an English sentence where the English language has specific rules that must be followed. If you see the sentence "The boook have many pages", you will immediately see 2 syntax errors i.e. the spelling of the word "book" and the word "have" should have been "has". These errors are syntax errors. They violate the rules of the English grammar.

Computer programs are written using computer languages. Computer languages also have specific rules that must be taken into consideration. Fortunately Visual Basic.NET programs are compiled, as you know by now, and the compiler will issue error messages for the programmer to fix. These errors are not difficult to correct.

7.2.2 LOGICAL ERRORS

Logical errors can sometimes be very difficult to correct. In everyday life it is not possible to leave your house before you are dressed. This is not the logical way to behave!

A very simple example of a logical error in a computer program is the following:

A program contains an instruction that asks the computer to add a variable called number to another variable called total. But the programmer has forgotten to ensure that the variable number has a valid value! The instructions could have been the following:

```
total = total + number
enter number
```

These 2 instructions are in the wrong order!

Different techniques will be discussed to assist the programmer to correct logical errors.

7.2.3 DATA ERRORS

Data errors are usually found in the input data. The techniques to handle these types of errors were discussed in the level 2 course where the input data was tested for correctness and errors were indicated.

7.2.4 RUN-TIME ERRORS

A run-time error occurs while the program is running. The cause of this type of error may be any of the above, but will seldom be a syntax error. A program is usually only executed fully when the program is free of any syntax errors.

A run-time error causes the program to produce incorrect results or stop abnormally.

7.3 TRACE TABLES

A trace table can be used to test the correctness of a program. Sometimes there may be only a small error in a program that is not obvious, but if a program is carefully checked, the chances are very good that the output will be correct.

Let us use an example to illustrate this concept.

Write an algorithm to calculate the product of 2 integers i.e. the variables num1 and num2. Store the answer in the variable product.

```
CalcProduct
        ~       this program will calculate the product of 2 integers

        display "enter the first integer"
        enter num1
        display "enter the second integer"
        enter num2
        product = num1 * num2
        display "The product is ", product
end
```

To do a trace table we need to draw a table with an instruction column, a column for each of the variables and a column for the output (which will be shown on the screen).

The instructions will be written in the first column in the same sequence as in the program.

To test the program, num1 = 4 and num2 = 5

instruction	num1	num2	product	output
1. display				enter the first integer
2. enter	4			
3. display				Enter the second integer
4. enter		5		
5. calculate			20	
6. display				The product is 20

This method is also called **desk checking**, because it is done even before the program is written in Visual Basic.NET.

Let us do a second example that contains selection statements to show how these types of statements should be included.

The programmer must plan a program to sell tickets to customers for a show. The price of a ticket is R30, but if a customer buys from 10 – 20 tickets, the price per ticket is only R28. The price for more than 20 tickets is R25 per ticket. The customer must enter the number of tickets he wants to buy and then the program must display the amount due.

We need to include another column to show the outcome of the if statement.

Algorithm:

BuyTickets

```
        display ”How many tickets do you want to buy?”
        enter noTickets
        if noTickets < 10
                amtDue = noTickets * 30
        else
                if noTickets < 20
                        amtDue = noTickets * 28
                else
                        amtDue = noTickets * 25
                endif
        endif
        display “The amount due for “, noTickets, “ tickets is R”, amtDue
end
```

We are going to test this algorithm by using 20 as the number of tickets.

instruction	noTickets	amtDue	Outcome of if	output
1. display				How many tickets do you want to buy?
2. enter	20			
3. if			False	
4. if			False	
5. calculation		500		
6. display				The amount due for 20 tickets is R500

This answer is incorrect!!!! When a person buys 20 tickets, the price is R28 per ticket and then the amount due must be R560.

We will study our if statements and will notice that the second if statement should be
if noTickets <= 20 instead of *if noTickets < 20*.

The correct algorithm will now be:
BuyTickets

```
        display ”How many tickets do you want to buy?”
        enter noTickets
        if noTickets < 10
                amtDue = noTickets * 30
        else
                if noTickets <= 20
                        amtDue = noTickets * 28
                else
                        amtDue = noTickets * 25
                endif
        endif
        display “The amount due for “, noTickets, “ tickets is R”, amtDue
end
```

You can redo the trace table using other values to test the algorithm to ensure that the program will produce the correct answers.

It is quite important to test your algorithm in this way before the program is written. You may save valuable time!



Exercises

Write the algorithms and then do the trace tables for each of the problems to test the algorithm for correctness.

1. Enter the temperature in degrees Fahrenheit (F) and convert it to degrees Celsius (C) using the following formula: $9C = 5(F - 32)$. Display the temperature in Fahrenheit and Celsius. Test the algorithm using 32^oF and 212^oF. (The answers should be 0^oC and 100^oC).

2. Enter two numbers. If the numbers are equal, increase the sum of the 2 numbers by 20%, but if the first number is less than the second number, calculate the difference between the two numbers. If the first number is greater than the second number, decrease the product of the two numbers by 25%. Store the answer in every case in the variable called *answer*. Display the value of *answer* on the screen. Test the algorithm with the numbers 12 and 5, 4 and 8, 7 and 7.

Calculate the answers beforehand in every case to ensure that your algorithm renders the correct answers.

7.4 DISPLAY VALUES STORED IN VARIABLES

It is possible that a program can render incorrect answers or even terminate abnormally.

If the program produces incorrect answers after calculations were done, the cause may be incorrect input data or incorrect calculations.

Examples of incorrect calculations:

- The annual salary is given and the monthly salary must be used to calculate the net salary. The annual salary was not divided by12 before calculations were done.
- The program should have calculated 5% of an amount and instead of multiplying by 0.05, the amount was multiplied by 5 or 0.5.
- An average point must be calculated, but the program did not increment a count; so the program tried to divide the total points by a count which is equal to zero.
- The program tried to use a variable in a calculation, but the variable does not contain any value or an invalid value.

There are many more of these types of errors and a method must be found to assist the programmer to correct all errors.

In the case of an incorrect result of the calculation where the number of items were multiplied by the unit price i.e.

```
amount =   noItems * price
```

the programmer will be advised to display the values of the number of items and the price before the multiplication statement by inserting the following statement:

```
display "no of items = ", noItems, " price = ", price
amount = noItems * price
```

The error could now be determined.

A display statement can also be inserted after the calculation because the receiving field (amount in this case) may not be of the correct type to store the answer.

Example:

Do the planning and write a VB.NET program to solve the following problem:

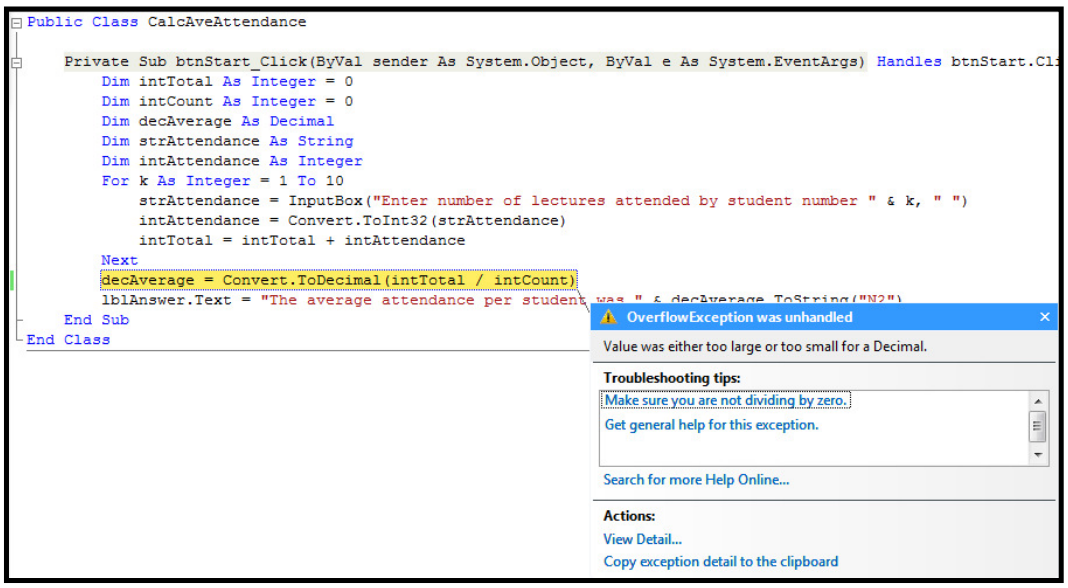
A group of 10 students enrolled for a course that consists of 15 lectures. As all the students did not attend all the lectures, the head of department wanted to know what the average attendance was. Enter the number of lectures attended by every student and then calculate and display the average.

Algorithm:

CalculateAverageAttendance

```
        total = 0
        count = 0
        for k = 1 to 10
                display "enter number of lectures attended by student number ",k
                enter attendance
                total = total + attendance
        next
        average = total / count
        display "The average attendance per student was ", average
end
```

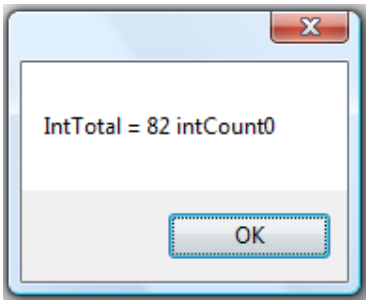
The programmer felt it was unnecessary and time wasting to do a trace table to test the correctness of his algorithm. He started programming immediately using the algorithm and then the program ended up in displaying the following error message.



The programmer can now display the values in intTotal and intCount, because it is clear that there must be a problem with one or both of these values and that would be why decAverage could not be calculated.

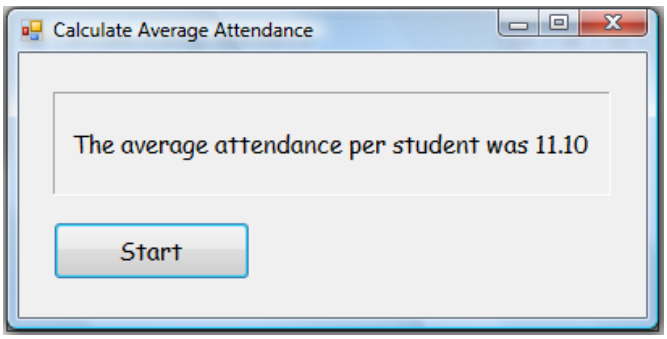
One way to display the values is by means of a message box.

```
MessageBox.Show("IntTotal = " & intTotal & " intCount" & intCount)
        decAverage = Convert.ToDecimal(intTotal / intCount)
        lblAnswer.Text = "The average attendance per student was " & _
                         decAverage.ToString("N2")
```



The message box indicated that intCount is still 0 and division by 0 is not allowed.

The programmer must either increment intCount during every execution of the loop or the total can be divided by 10 as we know that there are 10 students. If we fix this error, the following output will be displayed, based on the input values entered.





Practical

Write a Visual Basic.NET program to solve the following problem without doing any planning and include one or more errors. These errors can be omitting statements or doing an incorrect calculation. Run the program to see how it stops or what results are produced, and then insert statements to show the values or variables before calculations are done.

An unknown number of students are taking part in a sports day. They are grouped in groups A, B and C. The management needs to know how many students are in each group and also what percentage of the total is in a specific group. Display the results. The programmer must decide how to terminate the program. Hint: include incorrect test data, include an incorrect if statement and also an incorrect way of calculating the percentages.

7.5 INCLUDE STUBS IN A PROGRAM

A very useful way to ensure that a program runs correctly is to include stubs in a program. This means that instead of including a specific set of instructions, a fixed value is assigned to a variable just to ensure that the flow of the program is correct. Display statements may also be inserted in order for the programmer to see that all the statements are processed.

Students consider it unimportant to use this method, because they are sure that they can write a program without any errors and this method is just a waste of time. But, the programmer will find that a program written in this way will contain only a few errors, if any. Let us look at an example to explain this method:

Do the planning using stubs and write a VB.NET program for every phase to determine the average weight of baggage of a number of passengers on a flight.

```
CalculateAverageBaggage
        sum = 0
        count = 0
        baggage = 2
        do while baggage <> -1
                sum = 20              ~  constant is used
                count = 2
                baggage = -1          ~  program termination
        loop
        average =10                   ~  constant is used
        display "The average weight of the baggage is ", average
end
```

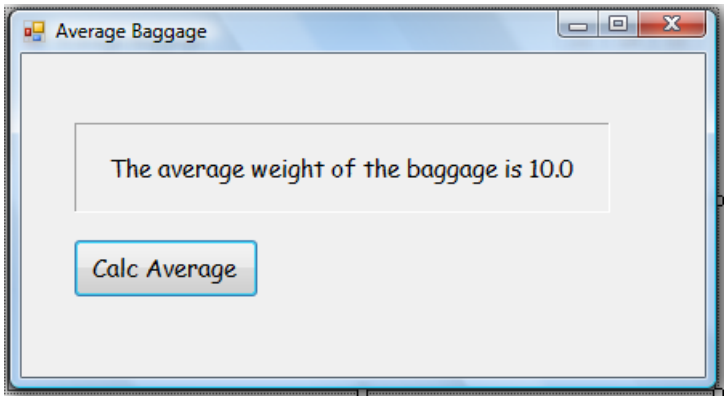
The above program does not contain any calculations, but constants have been inserted instead of the calculations. It is not possible to have any incorrect calculations in the program above.

Code in the code editor:

```
Public Class CalcAveBaggage

Private Sub btnAverage_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnAverage.Click
        Dim intSum = 0
        Dim intCount = 0
        Dim decAverage As Decimal
        Dim intBaggage As Integer
        intBaggage = 2
        Do While intBaggage <> -1
            intSum = 20
            intCount = 2
            intBaggage = -1
        Loop
        decAverage = 10
        lblAnswer.Text = "The average weight of the baggage is " & _
                         decAverage.ToString("N1")
    End Sub
End Class
```

It is clear that this program will produce output, although the result is not correct. It will be 10.0 every time. However, the flow of the program is tested and it is clear that the loop works and output is produced.



Now that it has been established that the program performs the correct logical sequence, it is now possible to include the correct logical construct code to perform the designated tasks.

Step 1: Input will be added to the program, but constants will still be used in the numeric fields. This is just to ensure that the flow of the program is correct.

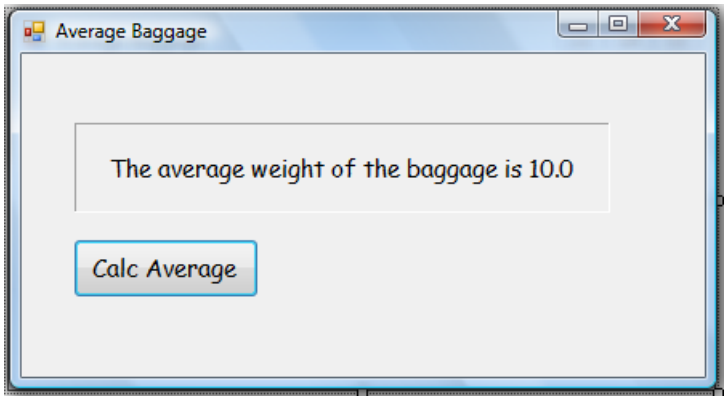
```
CalculateAverageBaggage
        sum = 0
        count = 0
        display "enter the weight of the first passenger's baggage, -1 to stop"
        enter baggage
        do while baggage <> -1
            sum = 20
            count = 2
            display "enter the weight of the first passenger's baggage, -1 to stop"
            enter baggage
        loop
        average = 10
        display "The average weight of the baggage is ", average
    end
```

Code in the code editor:

```
Public Class CalcAveBaggage

Private Sub btnAverage_Click(. . .) Handles btnAverage.Click
   Dim intSum = 0
   Dim intCount = 0
   Dim strBaggage As String
   Dim decAverage As Decimal
   Dim intBaggage As Integer
   strBaggage = InputBox("Enter weight of the 1st passenger's baggage", _
                         "Enter -1 to stop")
   Do While strBaggage <> "-1"
      Integer.TryParse(strBaggage, intBaggage)
      intSum = 20
      intCount = 2
      strBaggage = InputBox("Enter weight of next passenger's baggage", _
                            "Enter -1 to stop")
   Loop
   decAverage = 10
   lblAnswer.Text = "The average weight of the baggage is " & _
                    decAverage.ToString("N1")
End Sub
End Class
```

It is clear that the output will still be the same as we have not calculated the average as yet.



If this program is now running smoothly, the calculation statements may be inserted and the programmer can be sure that the program will work correctly (or almost work correctly!)

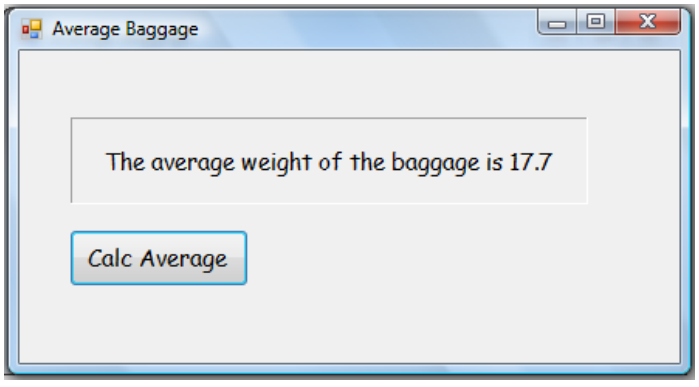
```
CalculateAverageBaggage
        sum = 0
        count = 0
        display "enter the weight of the first passenger's baggage, -1 to stop"
        enter baggage
        do while baggage <> -1
                sum = sum + baggage
                count = count + 1
                display "enter the weight of the first passenger's baggage, -1 to stop"
                enter baggage
        loop
        average = sum / count
        display "The average weight of the baggage is ", average
end
```

Complete program in the code editor:

```
Public Class CalcAveBaggage

Private Sub btnAverage_Click(. . .) Handles btnAverage.Click
   Dim intSum = 0
   Dim intCount = 0
   Dim strBaggage As String
   Dim decAverage As Decimal
   Dim intBaggage As Integer
   strBaggage = InputBox("Enter weight of 1st passenger's baggage", _
                         "Enter -1 to stop")
   Do While strBaggage <> "-1"
      Integer.TryParse(strBaggage, intBaggage)
      intSum = intSum + intBaggage
      intCount = intCount + 1
      strBaggage = InputBox("Enter weight of next passenger's baggage", _
                            "Enter -1 to stop")
   Loop
   decAverage = Convert.ToDecimal(intSum / intCount)
   lblAnswer.Text = "The average weight of the baggage is " & _
                    decAverage.ToString("N1")
End Sub
End Class
```

Output, based on input values entered:





Questions

What will happen if the enter statement at the end of the loop is omitted?

Another example:

Jack did some shopping and bought 5 different items. The user enters the price of every item and the total amount is calculated. Add 14% sales tax to the total amount and display the answer. The total amount must be calculated in the function CalcTotal and the sales tax is calculated in the function AddTax.

This is only a short program and the programmer can easily code it straight away, but it will illustrate the use of stubbing.

Firstly, only the skeleton of the program is written to test the flow of the logic.

```
Function CalcTotal ( )
        total = 10                          ~  insert a value for total
return total

Function AddTax( )
        finalTotal = 11.4                   ~  insert a value for finalTotal
return finalTotal

ShoppingProgram
        totalAmt = 0
        for p = 1 to 1                      ~  loop only once
                display "enter the price of the item
                price = 10                  ~ insert a value for price
                totalAmt = CalcTotal()
        next
        finalAmt = AddTax()
        display "Jack has to pay R", finalAmt
end
```

First phase of the program in VB.NET:

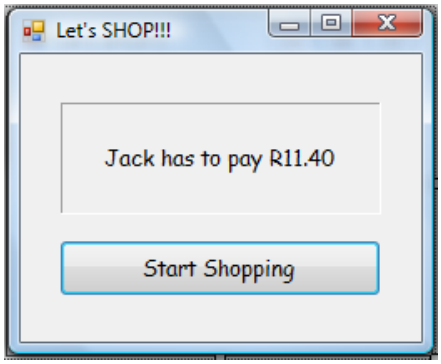
```
Public Class Shopping

Private Function CalcTotal() As Decimal
   Dim decTotal As Decimal = 10
   Return decTotal
End Function
```

```
Private Function AddTax() As Decimal
   Dim decFinalTotal As Decimal = 11.4
   Return decFinalTotal
End Function

Private Sub decShop_Click(. . .) Handles decShop.Click
   Dim decTotalAmt As Decimal = 0
   Dim decFinalAmt As Decimal
   Dim decPrice As Decimal
   For p As Integer = 1 To 1
       decPrice = 10
       decTotalAmt = CalcTotal()
   Next
   decFinalAmt = AddTax()
   lblAnswer.text = "Jack has to pay R" & decFinalamt.ToString("N2")
 End Sub
End Class
```

Output at this stage:



When the program above is working correctly, the program can now be expanded to include the calculations instead of the stubs and the loop can also be inserted to repeat the statements 5 times.

```
Function CalcTotal (price, total)
        total = total + price                    ~  calculate the total
return total

Function AddTax (total)
        finalTotal = total * 1.14                ~  add the sales tax
return finalTotal

ShoppingProgram
        totalAmt = 0
        for p = 1 to 5                      ~  loop 5 times
                display "enter the price of the item
                enter price
                totalAmt = CalcTotal(price,totalAmt)
        next
        finalAmt = AddTax(totalAmt)
        display "Jack has to pay R", finalAmt
end
```

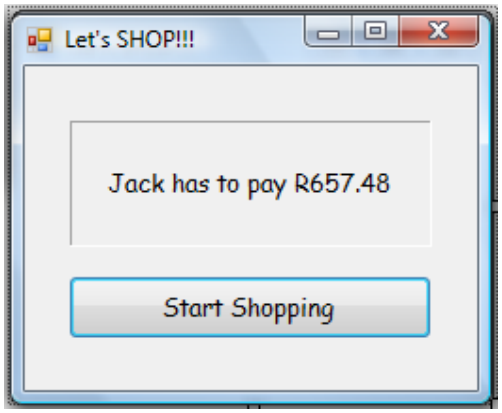
Final code in the code editor:

```
Public Class Shopping

Private Function CalcTotal(ByVal decPrice As Decimal, _
                           ByVal decTotal As Decimal) As Decimal
   decTotal = decTotal + decPrice
   Return decTotal
End Function

Private Function AddTax(ByVal decTotal As Decimal) As Decimal
   Dim decFinalTotal As Decimal
   decFinalTotal = decTotal * 1.14
   Return decFinalTotal
End Function

Private Sub decShop_Click(. . .) Handles decShop.Click
   Dim decTotalAmt As Decimal = 0
   Dim decFinalAmt As Decimal
   Dim strPrice As String
   Dim decPrice As Decimal
   For p As Integer = 1 To 5
       strPrice = InputBox("Enter the price of the item", " ")
       Decimal.TryParse(strPrice, decPrice)
       decTotalAmt = CalcTotal(decPrice, decTotalAmt)
   Next
   decFinalAmt = AddTax(decTotalAmt)
   lblAnswer.text = "Jack has to pay R" & decFinalamt.ToString("N2")
End Sub
End Class
```





Practical

Plan and write programs in at least 2 phases using stubs to solve the following problems:

1. The employees of the EXCELLENT COMPANY retire at the age of 60. Their pension depends on the number of years the employees worked for the company according to the following table:

Number of years worked	**Percentage of final salary per year worked**
1 - 5	1%
6 – 15	1.5%
16 years and more (max 35)	2.1%

   The user must enter the last monthly salary of the employee and the number of years he/she has worked for the company.

   The program must calculate and display the monthly pension of the employee.
   Hint: insert values in the nested if statement in the first phase.

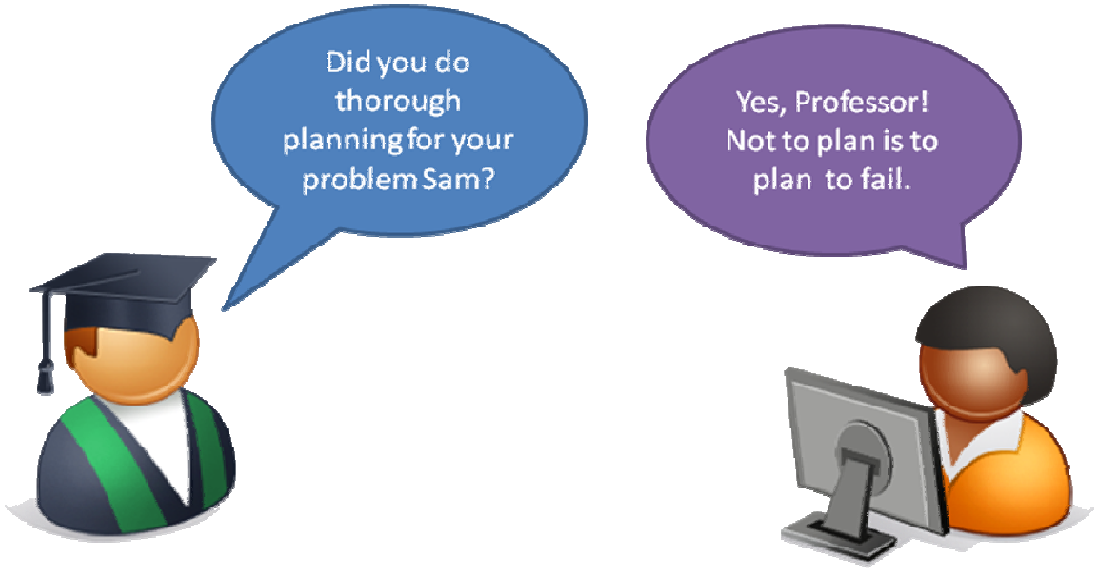
2. Write only a VB.NET program for the following:

   The user is requested to enter a number and may choose to do one of the following operations on the number and then display the answer:

   - Determine the number of digits in the number
   - Round the number to 1 decimal position
   - Calculate the square of the number

   Use a menu to display the different choices.

Questions and Answers



CHAPTER 1
QUESTIONS & ANSWERS
STRING MANIPULATION

QUESTIONS:

QUESTION 1

Indicate whether the following statements are TRUE or FALSE. Provide a reason for your answer.

1.1 `lblName.text = "My name is" & strName` will display the following on the label, if the variable strName contains the value Sue.
My name is Sue

1.2 If the .Length property is applied to a string, it will return a decimal value as an answer.

1.3 The following statement will result in an answer of 16.
```
intAnswer = "Word".Length ^ 2
```

1.4 If the value of strName is "Angela", `strName.IndexOf("n")` = 2.

1.5 `strNum.StartsWith("079")` will return a value of 1 if the string starts with 079.

1.6 The following statement is valid:
```
lstDisplay.Items.Add(strName.PadLeft(20) & decAmt.PadRight(20))
```

1.7 The following two statements are equivalent:
```
Amount = Amount.Trim("$"c, "%"c)
Amount = Amount.TrimLeft("$"c, "%"c)
```

1.8 When the variable strSentence contains the string "Today is a lovely day", the method `strSentence.IndexOf("day")` will return 18.

1.9 The Remove method can only remove one character at a time from a string.

1.10 It is compulsory to specify which character must be removed from the end of a string, when using the TrimEnd Method.

1.11 The following two statements will have the same effect:
```
If String.Compare(strValue1,strValue2) = -1 Then
If strValue1 < strValue2 Then
```

1.12 If strSentence contains the value "Today is a lovely day", the following two methods will both result in the answer "lovely day".
```
strSentence.Substring(11)
strSentence.Substring(11, 10)
```

1.13 The following two methods will result in the same answer:
```
If strName.StartsWith("C") Then
If strName.Substring(0, 1) = "C" Then
```

1.14 If strState contains the string "MI " (MI followed by 3 spaces), the following methods will result in an answer of 2.
```
(strState.Remove (2, 3).Length
```

1.15 The following two methods will result in the same answer:
```
If strNumber.EndsWith("9") Then
If strNumber.Substring(strNumber.Length, 1)= "9" Then
```

QUESTION 2

In each of the following cases, encircle the letter of the most correct answer.

2.1 The answer of `"These examples,".Length` is _____

a. 14
b. 15
c. 14.5
d. 15.5
e. None of the above

2.2 If `strLetters` contains the value "ABC123DEF", the value of `strLetters.IndexOf("c")` will be _____

a. 3
b. 4
c. 0
d. -1
e. None of the above

2.3 The following method is able to remove characters from a string.

a. Remove
b. Trim
c. TrimLeft
d. TrimRight
e. All of the above

2.4 Which of the following For-clauses can be used to access each character contained in the `strValue` variable, character by character? The variable contains 15 characters.

a. For intIndex As Integer = 0 to 15
b. For intIndex As Integer = 0 to strValue.Length – 1
c. For intIndex As Integer = 1 to strValue.Length
d. For intIndex As Integer = 1 to strValue.Length – 1
e. None of the above

2.5 Which of the following statements can be used to assign the seventh character in the variable `strWord`, to `strLetter`?

a. strLetter = strWord.Substring(6)
b. strLetter = strWord.Substr(7, 1)
c. strLetter = strWord.Substring(7, 1)
d. strLetter = strWord(7).Substring
e. None of the above

2.6 What will be returned by the method String.Compare("GAUTENG", "Gauteng")?

a. True
b. False
c. 0
d. -1
e. 1

2.7 Decimal numeric values can be right aligned by using the _____ method.

a. PadRight
b. PadLeft
c. RightAlign
d. Align
e. None of the above

2.8 Which of the following statements changes the contents of the variable `strPostalCode` from 1027 to 0027?

a. Mid (strPostalCode, 1) = "0"
b. Mid (strPostalCode, 1, 1) = "0"
c. strPostalCode = strPostalCode.Replace ("1", "0")
d. strPostalCode = strPostalCode.Remove (0, 1)
   strPostalCode = strPostalCode.Insert (0, "0")
e. All of the above

2.9 Which of the following statements changes the contents of the variable `strWord` from "red" to "read"

a. strWord = strword.Insert (2, "a")
b. strWord = strWord.Insert (3, "a")
c. strWord = strword.Replace (2, "a")
d. strWord = strword.AddTo (2, "a")
e. None of the above

2.10 If the variable `strName` contains the value "Rudy Ruthman", the method `strName.Contains("u")` will return _____

a. True
b. False
c. 1
d. 2
e. 0

2.11 The `strType` variable contains the word "A Types". Which of the following statements will change the contents of the variable to "A Type"?

a. strType = strType.Remove((strType.Length – 1), 1)
b. strType = strType.Replace("s"," ")
c. strType = strType.Substring(0, (strType.Length – 1))
d. strType = "A Type"
e. All of the above

2.12 Which of the following statements assigns the first 2 characters in the variable `strItem` to the variable called `strItemCode`?

a. strItemCode = strItem.Sub(0, 2)
b. strItemCode = strItem.Assign(0, 2)
c. strItemsCode = strItem.TrimLeft(0, 2)
d. strItemCode = strItem.Substring(0, 2)
e. strItemCode = strItem.Substring(1, 2)

2.13 Which of the following methods can be used to determine whether the variable `strNumber` ends with a 9?

a. strNumber = "9"
b. strNumber.Ends("9")
c. strnumber.Last = "9"
d. strNumber.EndsWith(9)
e. strNumber.EndsWith("9")

2.14 If `strName` = "Jennifer James", the statement
`Mid (strName, 14, 3) = "son"`
will result in:

a. Jennifer James
b. Jennifer Jameson
c. Jennifer Jamesson
d. Jennifer Jamson
e. None of the above

2.15 Which of the following statements will change the value of strDeptCode in the following way? The 5 characters contained in the variable strDeptCode must be combined with the number 123 and followed by asterisks. The variable must contain a total of 20 characters.

a. strDeptCode = (strDeptCode & "123").PadRight(20, "*")
b. strDeptCode & "123".PadRight(20, "*")
c. strDeptCode = strDeptCode & "123".PadRight(12, "*")
d. strDeptCode = (strDeptCode & 123).PadLeft(12, "*")
e. strDeptCode = (StrDeptCode & "123").PadLeft(20, "*")

QUESTION 3

Write string manipulation code for the following questions using the original given declarations for every question:

```
Dim strValueOne As String = "Charlie is a boy"
Dim strValueTwo As String = "Susie is a girl"
Dim strValueThree As String
Dim intNumber As Integer
```

a. Use the declarations to produce the sentence "Charlie is a boy and Susie is a girl" and store the answer in `strValueThree`.

b. Use the given declarations to produce the sentence "Charlie and Susie are children" and store the answer in `strValueThree`.

c. What is the difference in the length of `strValueOne` and `strValueTwo`? Store the answer in `intNumber`.

d. In which index position do we find the letter e in `strValueOne`?

QUESTION 4

Use the following declarations and apply the statements to find the value of `strAnswer`.

```
Dim strSentenceA As String = "English grammar lesson"
Dim strSentenceB As String = "prophet"
Dim strSubAnswer, strAnswer As String

strAnswer = strSentenceA.Remove(8, 8)
strAnswer = strAnswer.Replace("mar", "ming")
strSubAnswer = strSentenceB.Substring(0, 3)
strAnswer = strAnswer & " " & strSubAnswer
```

QUESTION 5

The first name followed by the last name of a student is stored in a string variable as follows: Peter Daniels. Write code to store the first name in a first name string variable and the second name in a last name string variable. Declare all the variables.

QUESTION 6

Given the following variable declarations:

```
Dim strDescription, strShortDescr As String
Dim intDescrIndex As Integer
```

Using the above declarations, write code to store the substring following the first s in the variable `strDescription` in `strShortDescr`. (It is possible that the variable description does not contain the letter s).

QUESTION 7

Solve the following problems by first doing the planning and then writing the VB.NET program.

7.1 Enter a series of telephone numbers. Every telephone number must be preceded by an area code followed by a space. The length of the actual telephone number must be 7 characters. If not, an appropriate error message must be displayed. For every valid entry, display the area code enclosed in brackets, followed by the telephone number. A hyphen must appear after the third character of the telephone number. Decide on a way to terminate the entering process.

7.2 An old Arabian legend has it that a fabulously wealthy but unthinking king agreed to give a beggar one cent and double the amount for 64 days. Using this information, write, run and test a VB.NET program that displays in a list box how much the king must pay the beggar on each of the 64 days. The amounts must be right aligned. (Use the .Padleft method to achieve it).

7.3 Many companies use a special digit, called a check digit when they assign account numbers to clients. Every time a transaction occurs, this digit is first validated in order to ensure that only valid transactions are processed. One method to assign a check digit to a number is the modulus 11 method. This method is illustrated by means of the following example:

Step 1: The digit on the extreme right-hand is multiplied by 2, the previous digit by 3, the previous one by 4 etc.
Step 2: The sum of these answers is determined
Step 3: The sum is divided by 11 and the remainder is determined
Step 4: If the remainder is 0, the check digit will be 0 otherwise the check digit will be 11 minus this remainder.

If the number is 208123, the check digit will be 7 and the number assigned to the client will therefore be 2081237. If a transaction occurs, the complete number is entered and the calculation is done on the first 6 digits to calculate the check digit and to compare it to the 7th digit.

	2	0	8	1	2	3
Times	7	6	5	4	3	2
Answers:	14	0	40	4	6	6

Sum: 14 + 0 + 40 + 4 + 6 + 6 = 70
Remainder: 70/11 = 6 remainder 4
Check digit: 11 – 4 =7

Plan and write a VB.NET program to enter a series of account numbers and to validate it. Every number must start with the values 10. Furthermore, every account number must be exactly 7 characters in length where the last digit is a control digit, calculated and assigned according to the modulus 11 method.

If correct, display a message to indicate that it is correct. Otherwise display a suitable error message to indicate the exact error that occurred. Decide on a method to terminate the process.

7.4 Enter all the identity numbers for employees working at the ABC-company. If the value of the 4 digits, from character 7 to 10 is greater or equal to 5000, this person is a male otherwise a female. Display the identity number of every person, followed by the gender. The identity numbers must be displayed with a space between character 6 and 7 and a space between characters 10 and 11. Decide on a way to indicate that the last identity number has been entered. After all identity numbers have been displayed, display the total number of male employees and the total number of female employees.

ANSWERS:

QUESTION 1

1.1 FALSE
There is no space at the end of the first string, therefore the value will be My name isSue

1.2 FALSE
The Length property will return an integer value.

1.3 TRUE
The length of "Word" is 4 and 4 ^ 2 = 16

1.4 FALSE
The value n is in position 2, but the index of position 2 = 1.

1.5 FALSE
The StartsWith method will return a Boolean value. In this case it will return a True.

1.6 FALSE
The PadRight method is only applicable to strings. Therefore, decAmt must first be converted to a string before the PadLeft Method can be applied to it. It must change as follows: `decAmt.ToString("N2").PadRight(20))`

1.7 FALSE
The trim method will remove all $'s and %-signs on both sides, while the TrimLeft method will only remove the $'s and %-signs at the beginning of the string.

1.8 FALSE
No beginIndex is specified and therefore it will start looking for the first occurrence of the substring "day" from index position 0. It will find it at index position 2.

1.9 FALSE
The number of characters to be removed may optionally be specified.

1.10 FALSE
If omitted, spaces will be removed.

1.11 TRUE
The compare method will return -1 if the first string is less than the second string.

1.12 TRUE
Both methods will start at index position 11 to find the substring. The first method will return all the characters from position 11 and the second method will return the next 10 characters starting at position 11. Yet, the remaining characters are only 10.

1.13 TRUE
Both methods will test is the first character is a C

1.14 TRUE
3 Characters will be removed, starting at index position 2. Therefore, the 3 spaces will be removed. The length of the remaining characters is 2.

1.15 FALSE
The second method tries to test if the last character in the string is a 9. Yet, the index of the last character is fount at index position Length – 1 because the first index is 0, not 1.

QUESTION 2

2.1 b
2.2 d
2.3 e
2.4 b
2.5 e
2.6 e
2.7 b
2.8 e
2.9 a
2.10 a
2.11 e
2.12 d
2.13 e
2.14 a
2.15 a

QUESTION 3

```
Dim strValueOne As String = "Charlie is a boy"
Dim strValueTwo As String = "Susie is a girl"
Dim strValueThree As String
Dim intNumber As Integer
```

a. Use the declarations to produce the sentence "Charlie is a boy and Susie is a girl" and store the answer in `strValueThree`.

```
strValueThree = strValueOne & " and " & strValueTwo
```

b. Use the given declarations to produce the sentence "Charlie and Susie are children" and store the answer in `strValueThree`.

```
strValueThree = strValueOne.Substring(0, 7) & " and " & _
          strValueTwo.Substring(0, 5) & " are children"
```

c. What is the difference in the length of `strValueOne` and `strValueTwo`? Store the answer in `intNumber`.

```
intNumber = strValueOne.Length - strValueTwo.Length
```

d. In which index position do we find the letter e in `strValueOne`?

```
strValueOne.IndexOf("e")
```

QUESTION 4

Use the following declarations and apply the statements to find the value of `strAnswer`.

```
Dim strSentenceA As String = "English grammar lesson"
Dim strSentenceB As String = "prophet"
Dim strSubAnswer, strAnswer As String

strAnswer = strSentenceA.Remove(8, 8)
The value of strAnswer will now be:   "English lesson"
strAnswer = strAnswer.Replace("mar", "ming")
No changes will appear
strSubAnswer = strSentenceB.Substring(0, 3)
strSubAnswer = pro
strAnswer = strAnswer & " " & strSubAnswer
English lesson pro
```

QUESTION 5

The first name followed by the last name of a student is stored in a string variable as follows: Peter Daniels. Write code to store the first name in a first name string variable and the second name in a last name string variable. Declare all the variables.

```
Dim strName As String = "Peter Daniels"
Dim strFirstName As String
Dim strSurname As String

strFirstName = strName.Substring(6)
strSurname = strName.substring(0, 5)

Take note: Other possible answers could also be correct
           E.g. the remove method can also be used.
```

QUESTION 6

Given the following variable declarations:

```
Dim strDescription, strShortDescr As String
Dim sintescrIndex As Integer
```

Using the above declarations, write code to store the substring following the first s in the variable `strDescription` in `strShortDescr`. (It is possible that the variable description does not contain the letter s).

```
intDescrIndex = strDescription.IndexOf("s")
If intDescrIndex = -1 Then
     MessageBox.Show("No s in the description")
Else
     intDescrIndex = intDescrIndex + 1
     strShortDescr = intDescription.Substring(intDescrIndex)
End If
```

QUESTION 7

7.1 Enter a series of telephone numbers. Every telephone number must be preceded by an area code followed by a space. The length of the actual telephone number must be 7 characters. If not, an appropriate error message must be displayed. For every valid entry, display the area code enclosed in brackets, followed by the telephone number. A hyphen must appear after the third character of the telephone number. Decide on a way to terminate the entering process.

Planning:

	Description	Type	Name of variable
Input:	telephone number	string	strNumber
Output:	new telephone number	string	strNewNumber

Algorithm:

TelNos

```
    display "enter the code and the telephone number"
    enter strNumber

    do while strNumber <> " "
        strNewNumber = strNumber.Insert (0, "(")  ~ insert first bracket
        intPosition = strNewNumber.IndexOf(" ")   ~ find position of space
        strNewNumber = strNewNumber.Insert(intPosition, ")")

        intPosition = intPosition + 2             ~  move to actual number
        if(strNewNumber.Length – intPosition) = 7
            intPosition = intPosition + 3         ~  determine hyphen position
            strNewNumber = strNewNumber.Insert(intPosition, "-")
            display "The number is ", strNewNumber
        else
            display "Error – number too long"
        endif
        display "enter the code and the telephone number"
        enter strNumber
    loop
end
```

Code in the code editor:

```
Public Class TelNos

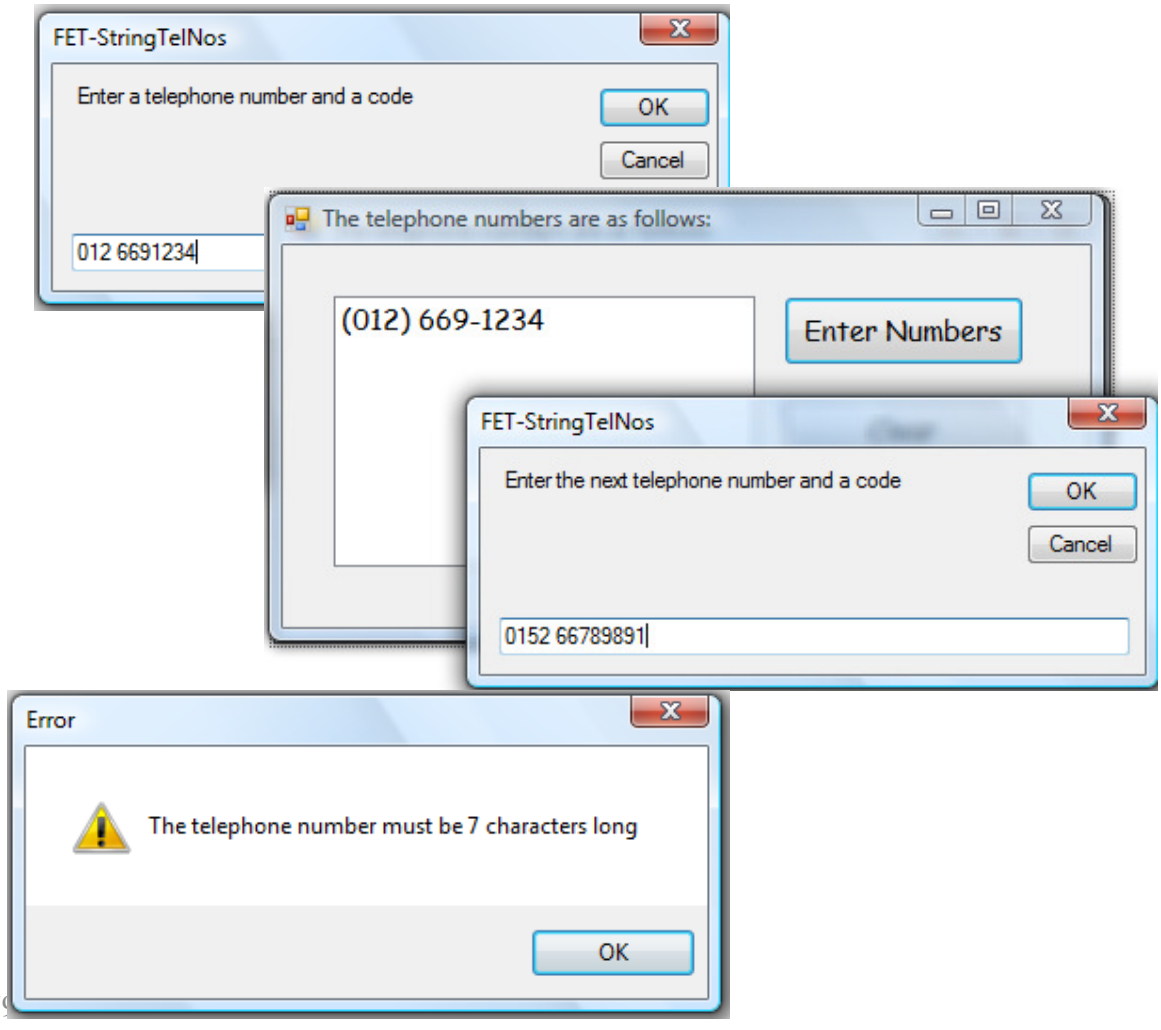
Private Sub btnEnter_Click(. . .) Handles btnEnter.Click
   Dim strNumber, strNewNumber As String
   Dim intPosition As Integer
   strNumber = InputBox("Enter a telephone number and a code")
   Do While strNumber <> ""
      strNewNumber = strNumber.Insert(0, "(")
      intPosition = strNewNumber.IndexOf(" ")
```

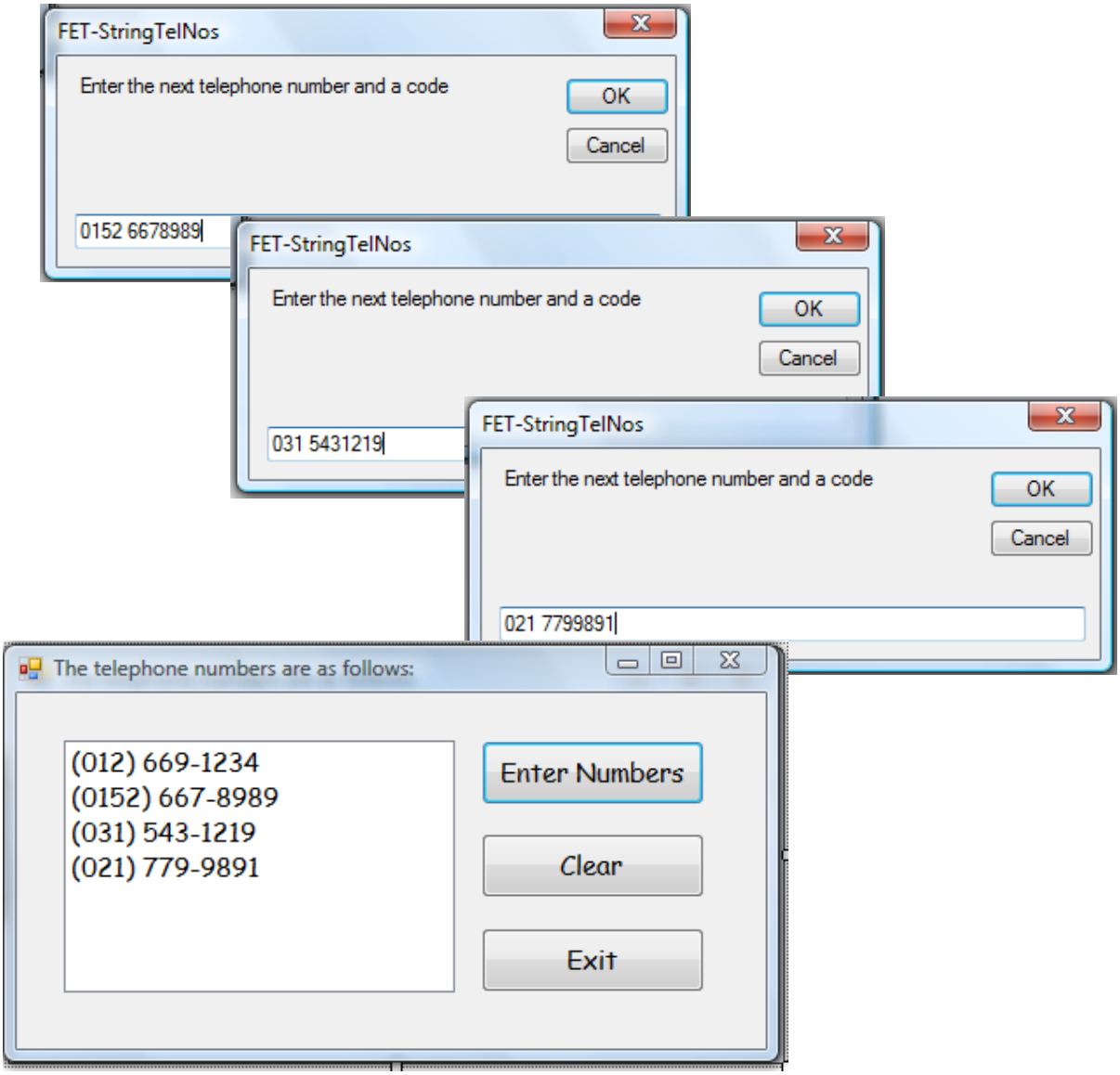
```
      strNewNumber = strNewNumber.Insert(intPosition, ")")
      'Determine position of actual telephone number
      'Position of ) plus 2
      intPosition = intPosition + 2
      If strNewNumber.Length - intPosition = 7 Then
         'Determine position where hyphen must be inserted
         intPosition = intPosition + 3
         strNewNumber = strNewNumber.Insert(intPosition, "-")
         lstDisplay.Items.Add(strNewNumber)
      Else
         MessageBox.Show("The telephone number must be 7 characters long",_
                "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
      End If
      strNumber = InputBox("Enter the next telephone number and a code")
   Loop
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   lstDisplay.Items.Clear()
End Sub

Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnExit.Click
   Close()
End Sub
End Class
```

Examples of output:





7.2 An old Arabian legend has it that a fabulously wealthy but unthinking king agreed to give a beggar one cent and double the amount for 64 days. Using this information, write, run and test a VB.NET program that displays in a list box how much the king must pay the beggar on each of the 64 days. The amounts must be right aligned. (Use the .Padleft method to achieve it).

Planning:

	Description	Type	Name of variable
Input:	none		
Output:	day number	integer	intDayNo
	amount	decimal	decAmt

Algorithm:

```
DisplayMoney
        BlnFirstTime = True                                  ~  set boolean value
        display "DAY", "       ", "AMOUNT TO PAY"            ~  write headings

        for intDay = 1 to 64
                if decAmount > 1000000 AND blnFirstTime – True
                        blnFirstTime = False
                        intDayNo = intDay     ~ the day the amount exceeded R1000000
                endif
                decAmt = decAmt * 2
        next
        display "On day ", intDayNo, " the king had to pay the beggar more than
                                                1 million rand!"
end
```

Code in the code editor:

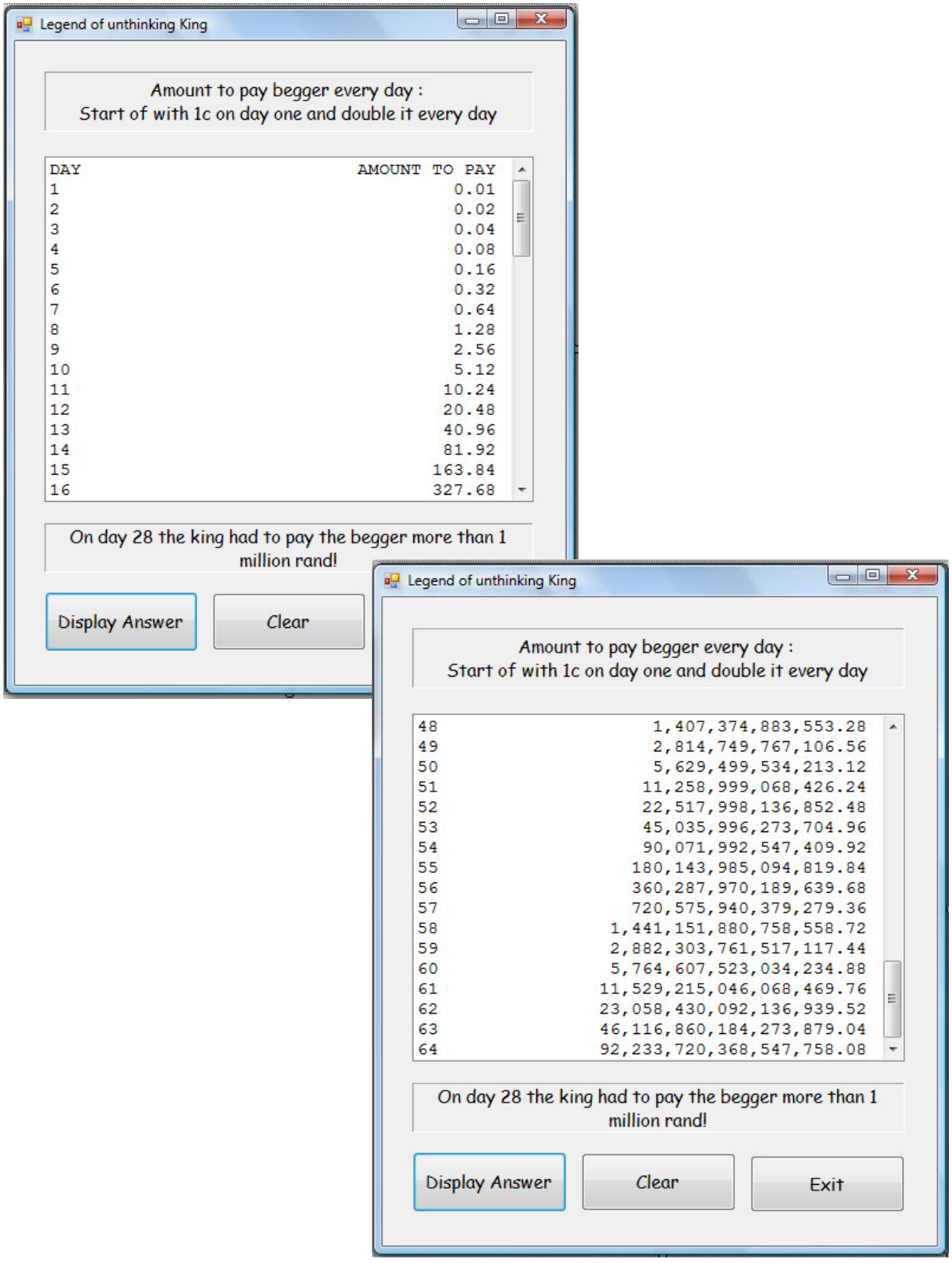
```
Public Class displayMoney.vb

Private Sub btnDisplay_Click(. . .) Handles btnDisplay.Click
   Dim decAmt As Decimal = 0.01
   Dim intDayNo As Integer
   Dim blnFirstTime As Boolean = True
   lstAnswer.Items.Add("DAY".PadRight(7) & "AMOUNT TO PAY".PadLeft(35))
   For intDay As Integer = 1 To 64
       lstAnswer.Items.Add(intDay.ToString.PadRight(7) & _
                           decAmt.ToString("N2").PadLeft(35))
       If decAmt > 1000000 And blnFirstTime = True Then
          blnFirstTime = False
          intDayNo = intDay
       End If
       decAmt = decAmt * 2
    Next
    lblInfo.Text = "On day " & intDayNo & _
          " the king had to pay the begger more than 1 million rand!"
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
    lstAnswer.Items.Clear()
    lblInfo.Text = ""
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
    Close()
End Sub
End Class
```

Example of output:



7.3 Many companies use a special digit, called a check digit when they assign account numbers to clients. Every time that a transaction occurs, this digit is first validated in order to ensure that only valid transactions are processed. One method to assign a check digit to a number is the modulus 11 method. This method is illustrated by means of the following example:

Step 1: The digit on the extreme right-hand is multiplied by 2, the previous digit by 3, the previous one by 4 etc.
Step 2: The sum of these answers is determined
Step 3: The sum is divided by 11 and the remainder is determined
Step 4: If the remainder is 0, the check digit will be 0 otherwise the check digit will be 11 minus this remainder.

If the number is 208123, the check digit will be 7 and the number assigned to the client will therefore be 2081237. If a transaction occurs, the complete number is entered and the calculation is done on the first 6 digits to calculate the check digit and to compare it to the 7th digit.

	2	0	8	1	2	3
Times	7	6	5	4	3	2
Answers:	14	0	40	4	6	6

Sum: 14 + 0 + 40 + 4 + 6 + 6 = 70
Remainder: 70/11 = 6 remainder 4
Check digit: 11-4 =7

Plan and write a VB.NET program to enter a series of account numbers and to validate it. Every number must start with the values 10. Furthermore, every account number must be exactly 7 characters in length where the last digit is a control digit, calculated and assigned according to the modulus 11 method.

If correct, display a message to indicate that it is correct. Otherwise display a suitable error message to indicate the exact error that occurred. Decide on a method to terminate the process.

Algorithm:

```
CheckDigit
        display "Enter the account number"
        enter strAccNumber

        dowhile strAccNumber <> " "
          intLength = strAccNumber.Length
            intSum = 0                  ~ initalize sum for new number
            if intLength = 7    ~ test valid length
              if strAccNumber.StartsWith ("10")
                intMultiply = 7         ~ for leftmost character
                for intPosition = 0 to (intLenght – 2)
                                ~  convert string to integer
                  intValue = strAccNumber.Substring(intPosition,1) * intMultiply
                  intSum = intSum + intValue        ~  add to sum
                  intMultiply = intMultiply – 1
                next
```

```
                    intRem = intSum mod 11     ~ get remainder
                    if intRem = 0
                      intCheckDigit = 0
                    else
                      intCheckDigit = 11 – intRem
                    endif

            ~       compare check digit to last digit in acc no
                    if intCheckDigit = strAccNumber.Substring(intLength – 1, 1)
                      strMessage =  "This account number is valid"
                    else
                      strMessage = "Invalid account no – the last digit must be ",
                                                          intCheckDigit
                    endif
                  else
                    strMessage = "The account number must start with 10"
                  endif
                else
                  strMessage =  "The account number must be exactly 7 characters"
                endif
                display strAccNumber, "    ", strMessage

            display "Enter the account number"
            enter strAccNumber
      loop
end
```

Code in the code editor:

```
Public Class CheckDigit

Private Sub btnEnter_Click(. . .) Handles btnEnter.Click

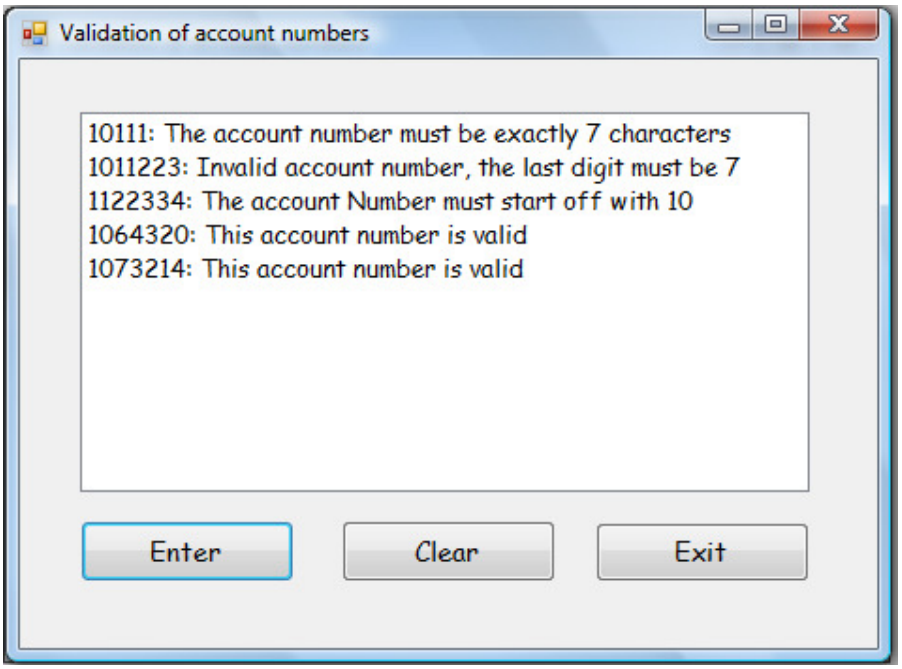
   'Declare variables
   Dim strAccNumber, strMessage As String
   Dim intPosition As Integer
   Dim intValue As Integer
   Dim intMultiply As Integer
   Dim intLength As Integer
   Dim intSum As Integer = 0
   Dim intRem As Integer
   Dim intCheckDigit As Integer
   Dim blnValid As Boolean = False
   'Enter first account number
   strAccNumber = InputBox("Enter the account number", "Only 7 character")
   'Loop until cancel-button is clicked
   Do While strAccNumber <> ""
      'Determine length of account number
      intLength = strAccNumber.Length
      'Initialize the sum. For every new number, the sum must start at 0
      intSum = 0
      'Validate the length, if invalid, display error message
```

```
      If intLength = 7 Then
         'Test if the number stats with 10, if not, display error message
         If strAccNumber.StartsWith("10") Then
            'Determine no with which to multiply digit on most left hand
            'side
            intMultiply = 7
            'Loop through all positions from most left hand side to second
            'last digit
            For intPosition = 0 To (intLength - 2)
                'Get the digit separately, multiply it and add it to the
                'sum
                intValue = _
                Convert.ToInt32(strAccNumber.Substring(intPosition, 1))_
                * intMultiply
                intsum = intsum + intValue
                'If first digit was multiplied by 7, the next one must be
                'multiplied by 6 etc.
                intMultiply = intMultiply - 1
             Next
             'Determine the remainder and check digit
             intRem = intsum Mod 11
             If intRem = 0 Then
                intCheckDigit = 0
             Else
                intCheckDigit = 11 - intRem
             End If
             'Compare the check digit with the last digit
             If intCheckDigit = _
                strAccNumber.Substring(intLength - 1, 1) Then
                strMessage = "This account number is valid"
             Else
                strMessage = _
                    "Invalid account number, the last digit must be " & _
                    intCheckDigit
             End If
         Else
            strMessage = "The account Number must start off with 10"
         End If
      Else
         strMessage = "The account number must be exactly 7 characters"
      End If
      lstAnswer.Items.Add(strAccNumber & ": " & strMessage)
      strAccNumber = InputBox(_
                  "Please enter next account number or cancel to stop", _
                  "Only 7 characters")
   Loop
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   lstAnswer.Items.Clear()
End Sub

Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnExit.Click
   Close()
End Sub
End Class
```

Examples of output:



7.4 Enter all the identity numbers for employees working at the ABC-company. If the value of the 4 digits, from character 7 to 10 is greater or equal to 5000, this person is a male otherwise a female. Display the identity number of every person, followed by the gender. The identity numbers must be displayed with a space between character 6 and 7 and a space between characters 10 and 11. Decide on a way to indicate that the last identity number has been entered. After all identity numbers have been displayed, display the total number of male employees and the total number of female employees.

Planning:

	Description	Type	Name of variable
Input:	identity number	string	strIdNumber
Output:	new identity number	string	strNewIdNumber
	gender	string	strGender
	total males	integer	intMaleTot
	total females	integer	intFemaleTot

Algorithm:

IdGender

```
        display "Enter the id number for the first employee"
        enter strIdNumber

        do while strIdNumber <> "  "
                intFourDigits = strIdNumber.Substring(6,4)  ~ find four numbers to test
                if intFourDigits >= 5000
                        strGender = "Male"
                        intMaleTot = intMaleTot + 1
                else
                        strGender = "Female"
                        intFemaleTot = intFemaleTot + 1
                endif

                strNewIdNumber = strIdNumber.Insert(10, " ")      ~ insert space
                strNewIdNumber = strNewIdNumber.Insert(6, " ")
                display strNewIdNumber, "   ", strGender

                display "Enter the id number for employee"
                enter strIdNumber
        loop

        display "Total number of male employees ", intMaleTot
        display "Total number of female employees  ", intFemaleTot
end
```

Code in the code editor:

```
Public Class IdGender

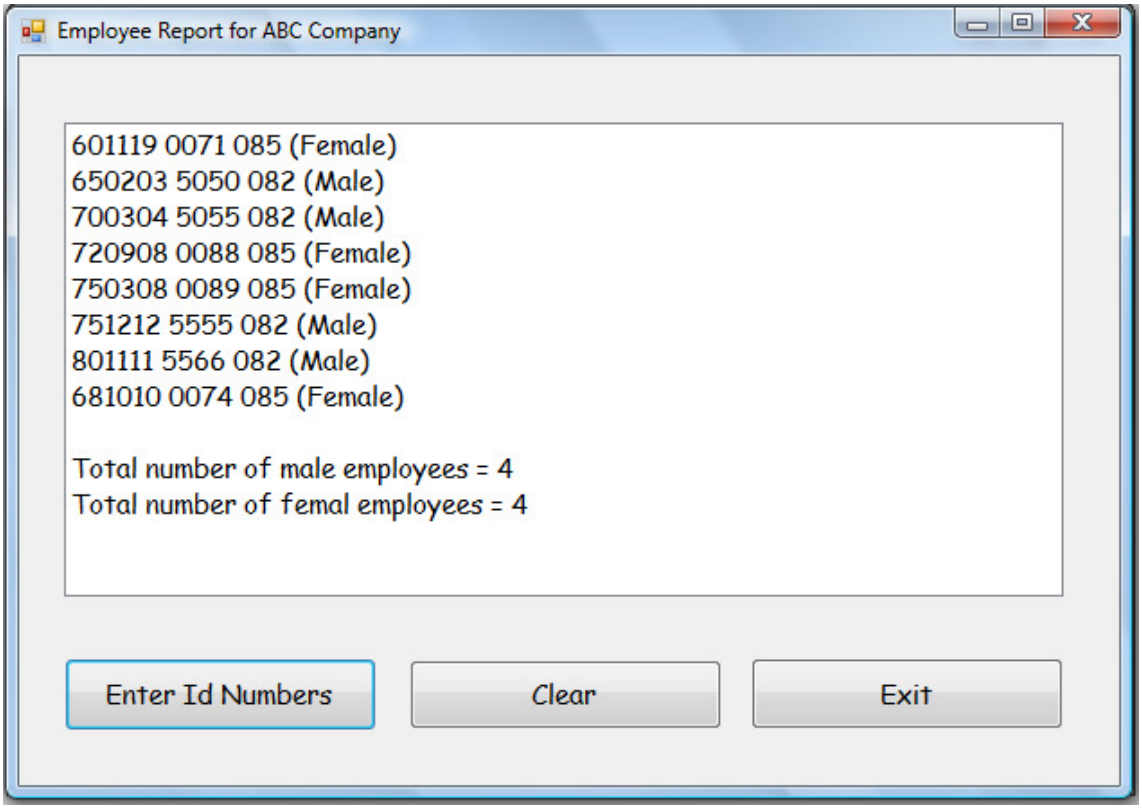
Private Sub btnEnter_Click(. . .) Handles btnEnter.Click
   'Id number entered
   Dim strIdNumber As String
   'New id number with spaces - to be displayed
   Dim strNewIdNumber As String
   'Variable to test whether the person is a male or female
   Dim intFourDigits As Integer
   'Gender to be displayed
   Dim strGender As String
   'Declare counters
   Dim intMaleTot As Integer = 0
   Dim intFemaleTot As Integer = 0
   'enter the first id number
   strIdNumber = InputBox("Enter the id number for the first employee",_
                          "Click cancel to stop")
   'Loop for all employees
   Do While strIdNumber <> ""
      'Find the 4 characters in the id-number, convert it to an integer
      intFourDigits = Convert.ToInt32(strIdNumber.Substring(6, 4))
      'Test if the person is male or female
```

```
      'display gender and count accordingly
      If intFourDigits >= 5000 Then
         strGender = " (Male)"
         intMaleTot = intMaleTot + 1
      Else
         strGender = " (Female)"
         intFemaleTot = intFemaleTot + 1
      End If
      strNewIdNumber = strIdNumber.Insert(10, " ")
      strNewIdNumber = strNewIdNumber.Insert(6, " ")
      lstDisplay.Items.Add(strNewIdNumber & strGender)
      strIdNumber = InputBox("Enter the id number for employee",_
                             "Click cancel to stop")
   Loop
   'display total number of male and female employees
   lstDisplay.Items.Add(" ")
   lstDisplay.Items.Add("Total number of male employees = " & _
                        intMaleTot.ToString)
   lstDisplay.Items.Add("Total number of femal employees = " & _
                        intFemaleTot.ToString)
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   lstDisplay.Items.Clear()
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Examples of output:



CHAPTER 2
QUESTIONS & ANSWERS
FUNCTION PROCEDURES AND SUB PROCEDURES

QUESTIONS:

QUESTION 1

Indicate whether the following statements are TRUE or FALSE. Provide a reason for your answer.

1.1 A function may return two answers to the calling module.
1.2 A sub procedure may calculate 2 answers and place it in reference parameters.
1.3 The parameter list in a sub procedure header may be optional.
1.4 The arguments in a function call must contain an indication of whether the parameters are sent by value or by reference.
1.5 A function call may contain 2 arguments and the corresponding function header may contain 3 parameters.
1.6 If the argument in the function call contains the hours as a decimal value, the corresponding parameter in the parameter list of the function header, may contain the hours as an integer value.
1.7 A function may call another function.
1.8 A sub procedure may not call another sub procedure.
1.9 There are two types of sub procedures in Visual Basic.NET.
1.10 If the variable is passed to a procedure by value, then the procedure has no access to this variable in memory.
1.11 You can tell if a variable is passed by value or by reference by looking at the arguments in the call statement.
1.12 The ByRef keyword tells the computer to pass the variable's address rather than its contents.
1.13 The ByRef keyword should appear after the parameter's name.
1.14 Every parameter in the function header must contain a data type.
1.15 The data type of the expression in the Return statement may only be integer or decimal.

QUESTION 2

In each of the following cases, encircle the letter of the most correct answer.

2.1 Which of the following is true about a sub procedure

a. It may receive only value parameters
b. It may receive only reference parameters
c. It need not receive any parameters
d. It may receive both value and reference parameters
e. All of the above

2.2 Which one of the following statements is true

a. A function may call another function.
b. A procedure may be used to input values for the variables named *quantity* and *price*.
c. A procedure may be used to change the value of more than one variable
d. All of the above
e. Only answers b and c.

2.3 Which one of the following statements is true about a function:

a. A function may receive a maximum of 3 value parameters
b. The first line of a function contains only the name of the function.
c. The addresses of the value parameters are included in the name of the parameters.
d. A function may return two values.
e. A function returns only one value.

2.4 A reference parameter

a. contains the address of a variable
b. is equivalent to a constant
c. contains a copy of the value of the variable
d. is all of the above
e. is none of the above

2.5 Study the following function call in an algorithm:

```
GrossPay = CalculatePay(Hours, Rate)
```

Which of the following function headers correspond to this call?

a. Function Calculate(valRate, valHours)
b. Function CalculatePay(valHours, valRate, refGrossPay)
c. Function CalculatePay(refHours, refRate)
d. Function CalculatePay(valHours, valRate)
e. Function CalculatePay(valRate, valHours)

2.6 Study the following procedure header in an algorithm:

```
Sub Rectangle (valLength, valWidth, refArea, refCircumference)
```

Which of the following will be a valid statement to call this procedure?

a. Call Rectangle (valLength, valWidth, refArea, refCircumference)
b. Call Rectangle (Length, Width, Area, Circum)
c. Area = Rectangle(Length, Width, Circum)
d. Call Rectangle (Width, Length, Area, Circumference)
e. Call Rectangle (Length, Width, Circum, Area)

2.7 Variables listed in a Call-statement are referred to as _____:

a. arguments
b. parameters
c. passers
d. all of the above
e. none of the above

2.8 In a procedure header, what comes after the sub keyword?

a. return value
b. procedure name
c. parameter list
d. end keyword
e. none of the above

2.9 The keyword to invoke an independent sub procedure in VB.NET is _____

a. Invoke
b. Call
c. Sub
d. Trigger
e. none of the above

2.10 Data passed to a sub procedure can be _____

a. By Value
b. By Reference
c. By Address
d. By Value or By Reference
e. none of the above

2.11 Which of the following function headers in VB.NET is not valid

a. Private Function CalcTax() As Decimal
b. Private Function CalcTax(Salary) As Decimal
c. Private Function CalcTax(ByVal Salary As Decimal) As Decimal
d. Private Function CalcTax(ByVal Salary As Decimal)
e. b and d

2.12 Which one of the following statements is true about a reference parameter?

a. it may be used in a function header
b. it may be used in a procedure header
c. it is also referred to as value parameters
d. if the module changes this parameter during execution time, it affects the calling module
e. b and d

2.13 Which one of the following statements is true about a value parameter?

a. it may be used in a function header
b. it may be used in a procedure header
c. the function may change the copy of the original variable during execution time
d. all of the above
e. none of the above

2.14 To determine whether a variable is being passed by value or by reference, you will need to examine _____

a. the call statement
b. the procedure header
c. the statements in the procedure
d. all of the above
e. either a or b

2.15 Which of the following function headers are valid if the function receives an integer value and must test whether it is a multiple of 3? If it is a multiple of 3, the function must return the value TRUE, else it must return the value FALSE.

a. Private Function Multiple3(ByRef intNum As Integer) As String
b. Private Function Multiple3(ByVal intNum As Integer) As String
c. Private Function Multiple3(ByVal intNum As Integer)
d. Private Function Multiple3(ByVal intNum As Decimal) As Boolean
e. Private Function Multiple3(ByVal intNum As Integer) As Boolean

QUESTION 3

A painter wants to know how much paint is needed to paint only the walls of the inside of a room. The room contains 2 windows and one door. The 2 windows are of the same size. The chosen paint covers 15 square meter per litre. You may assume that all variables in the main procedure have been declared and contain valid values. Study the following function and sub procedure calls in the main procedure. Write the corresponding function and sub procedures in VB.NET.

```
decAreaRoom = CalcArea1(decLengthWall, decWidthWall, decHeightWall)
decAreaWindow = CalcArea2(decLengthWindow, decHeightWindow)
decAreaDoor = CalcArea2(decHeightDoor, decWidthDoor)
Call CalcFinalAreaAndPaint(decAreaRoom, decAreaWindow, decAreaDoor, decPaint)
lblAnswer.Text = "The total area to be painted is "  & decAreaRoom.ToString("N1") _
        & ControlChars.Newline & decPaint.toString("N1") & _
        " litres paint will be needed to paint the room"
```

QUESTION 4

In each of the following cases, write the function call and function in VB.NET for the following:

4.1 Danny buys a number of items at R10 per packet of 8 but pays R1.50 for a single item. Calculate the smallest amount (decAmtDue) he has to pay by calling a function. He buys an unknown number of items (intNoItems) and the function must receive this value as a parameter.

4.2 Call a function to determine if a student will have examination entrance or not. Display either the message "Congratulations, you have examination entrance" or "Sorry, unfortunately you don't have examination entrance" on the label called lblOutcome, based on the value returned. The student will have examination entrance if both the following conditions are true: The student has a predicate of at least 40% (stored in intPredicate) and his/her class attendance is at least 85% (stored in intAttendance). The function must receive both these variables as parameters.

4.3 Call a function to calculate and return the number of pages needed for a copy-job (intNoPages). It depends on the number of pages in the master document (intMasterPages) and the number of copies to be made (intNoCopies). A Boolean variable (blnBackToBack) will indicate whether the copies must be made back to back or not. Provide for 10% additional paper due to the possibility of paper jams. The function must receive all the necessary parameters in order to do the calculation.

QUESTION 5

Predict the output for the following code:

```
Private Function CalcA (ByVal intA As Integer, ByVal intB As
integer) As Decimal
     Return Convert.ToDecimal(intA + intB / 2)
End Function

Private Sub CalcB (ByVal intA As Integer, ByRef intB As Integer,
ByRef decC As Decimal)
     Dim intX As Integer
     decC = intA \ intB
     intB = intB - intA
     intA = intA + 15
     For intX = 1 to 3
          intB = intB + 2
     Next
End Sub

Private Sub btnCalc_Click( . . .) Handles btnCalc.Click
     Dim intX As Integer = 1
     Dim intY As Integer = 5
     Dim decZ As Decimal = 0
     decZ = CalcA(intX, intY)
     lblAnswer1.Text = decZ.ToString("N1")
     decZ = CalcA(6,200)
     lblAnswer2.Text = decZ.ToString("F0")
```

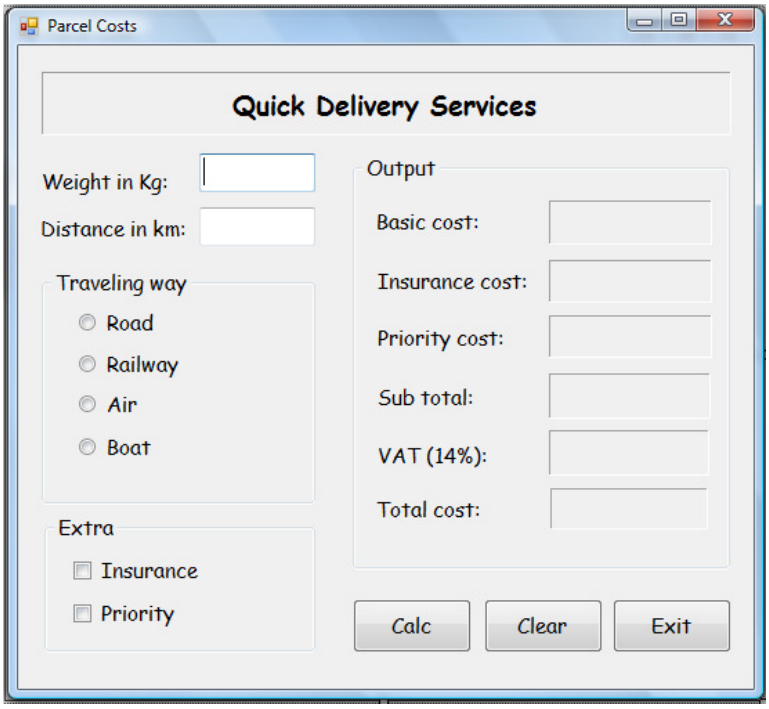
```
        Call CalcB(intX, intY, decZ)
        lblAnswer3.Text = "intX = " & intX & ControlChars.Newline & _
                          "intY = " & intY & ControlChars.Newline & _
                          "decZ = " & decZ.ToString("N")
End Sub
```

QUESTION 6

Solve the following problems by first doing the planning and then writing the VB.NET program.

6.1 At the Cheap Unfair Store, salaries for employees are computed as follows: A male employee earns a basic salary of R2500 per month. He receives an additional R250 for every year of experience. He also receives an additional R550 if he has a qualification. Males pay 10% tax and 8% pension fund, which is deducted from the gross salary to give his net salary. Female employees earn a basic salary of R2000 per month and an additional R200 for every year of experience. Females receive an additional R450 if they have a qualification and they pay 9.5% tax and 7.5% pension fund. Enter the gender, years experience and an indication if the person has a qualification or not. Call a separate function for females and males, which must each return the gross monthly salary. Then call a procedure to calculate the tax amount and pension fund amount. Call another function to return the net salary for this person. Display the gross and net salaries in the main module. You may assume that the input values will be valid.

6.2 In chapter 2, in 2.5.1, the following user interface and example are used to demonstrate the call of a sub procedure without parameters when the controls are cleared.



Plan and write the calculate button for this application where parcel costs are calculated. The basic cost of a parcel is determined by the weight of the parcel, the way the parcel will be sent as well as the travelling distance. The basic cost is calculated as R1.23 per kg multiplied by the distance and the travelling cost according to the tariffs in the following table:

Road	R0.15
Railway	R0.12
Air	R0.36
Boat	R0.25

Call a function to determine the travelling cost and then call a second function to calculate and return the basic cost. Call a sub procedure to calculate the insurance cost and priority costs. If the client chooses that the parcel must be insured, the insurance cost will be 11% of the basic cost. If the client indicates that the parcel is a priority, an additional 15% will be added to the basic cost. Call a function to calculate the sub total at this stage (basic cost + insurance cost + priority cost). The main module must then calculate and add 14% VAT to the sub total to calculate the total cost of the parcel. A sub procedure must be called to display all answers.

6.3 *TREAT YOURSELF* is a guesthouse with 20 rooms that can all be utilized as either a single or a double room. The tariff for a single room is R250.50 per night and the tariff for a double room is R400.90 for two persons sharing. The prices include breakfast.

Your program must produce the following information at the beginning of every week.

- Bookings for every night (number of single and number of double rooms),
- Number of people to expect for breakfast the following day,
- Income for the day

The total income for the week must also be calculated and displayed.

Display the number of single rooms, number of double rooms, number of breakfasts as well as the expected income for every day of the week. This program must provide for all possible input errors. E.g. If the sum of the number of single rooms and number of double rooms exceed 20, an appropriate error message must be displayed and the user must be able to rectify the mistake.

Use a list box on your user interface to display the results and make use of the PadLeft and PadRight methods in order to display the output in columns.

Provide for the following functions and independent sub procedures:

- A **sub procedure, InputData**, to enter the number of single room bookings and the number of double room bookings for the night. The total number of rooms may not be more than 20. In such a case, a suitable error message must be displayed and the user must re-enter the values until the input data is correct.
- A **function, NumberForBreakfast**, to calculate and return the number of people to expect for breakfast the following morning,
- A **function, IncomeOneNight**, to receive the number of single room bookings and the number of double room bookings and to return the income for that night,
- A **sub procedure, CalcTotalIncome**, to accumulate the total income for the week.

The information for every night (Day number, number of single room bookings, number of double room bookings, number of people to expect for breakfast and income for the night) must be displayed in the main algorithm. The total income for the week must also be displayed in the main algorithm after the loop (for-loop for 7 nights) has terminated.

6.4 TEN TRUCKS TRANSPORT is a transport company that delivers parcels to various destinations. The company has trucks with a capacity of 500kg, 750kg, 1000 kg and 1250 kg. The majority of the trucks have a capacity of 500 kg.

The cost of a parcel depends on the distance to the destination as well as the weight of the parcel and customers pay R1 per km as well as R1.23 per kg of the parcel weight.

Before a truck departs, the distance to the destination must be entered as an integer value. The capacity of the truck must also be selected. The weights of all the parcels must then be entered and after every parcel weight has been entered, the amount payable must be calculated and the parcel number, weight and amount payable must be displayed.

Parcels may be added as long as the current load plus the parcel weight is within the capacity of the truck. The truck need not be full on departure and the entering process can therefore be terminated before the truck is full. It is also possible that one parcel could be too heavy and will cause that the capacity of the truck is exceeded, in which case a parcel of acceptable weight may replace it.

When the weight of all the parcels has been entered, the total amount due for the load must be displayed. A message indicating whether the truck is completely full or not is also displayed. If the truck is not completely full, the available space (in kg) must be displayed.

The program must provide for all possible input errors, in which case a suitable error message must be displayed.

Call a function to determine the capacity of the truck and call a sub procedure to handle all calculations for a valid parcel to be loaded onto the truck.

ANSWERS:

QUESTION 1

1.1 FALSE
A function may return only one answer because it returns the answer via the name of the function.

1.2 TRUE
If the 2 answers are placed in reference parameters, both the calling module and the sub procedure refers to the same address in memory and therefore these answers will be available to the calling module.

1.3 TRUE
It is possible that a sub procedure may receive no parameters. It could be possible that the sub procedure makes use of global variables or the sub procedure could fulfil a purpose where no variables are needed (such as clearing controls on the user interface).

1.4 FALSE
The arguments in a function are only the variable names. The corresponding parameters in the function header will have to indicate whether it has been sent by value or by reference.

1.5 FALSE
The number of variables in the argument list must correspond to the number of parameters in the function header.

1.6 FALSE
The data type of an argument and corresponding parameter must be the same.

1.7 TRUE
Functions may be nested.

1.8 FALSE
Sub procedures may also be nested.

1.9 TRUE
Visual Basic.NET has an independent sub procedure that must be called from another part of the program, but also have event procedures that is associated with a specific object and event. It will automatically be executed when the event occurs.

1.10 TRUE
A value parameter makes a copy of the value at a different memory location. The function or sub procedure will refer to this copy and the calling module will refer to the original value.

1.11 FALSE
Arguments in the call statement contain only the names of the variables that are passed to the function or sub procedure. The corresponding parameters must contain the indication whether it is passed by value or reference.

1.12 TRUE
ByRef indicates that the parameter is passed by reference and therefore the address of the variable is passed in order for both the calling module and called module to refer to its contents.

1.13 FALSE
The ByRef keyword should appear before the parameter's name in the parameter list of the header of the sub procedure.

1.14 TRUE

It must contain a data type and it must be the same as the data type of the corresponding variable in the argument list.

1.15 FALSE
The data type of the expression in the Return statement may only be of any data type e.g. string.

QUESTION 2

2.1 e
2.2 d
2.3 e
2.4 a
2.5 d
2.6 b
2.7 a
2.8 b
2.9 b
2.10 d
2.11 e
2.12 e
2.13 d
2.14 b
2.15 e

QUESTION 3

```
Private Function CalcArea1 (ByVal decLength As Decimal, ByVal decWidth As Decimal, _
                            ByVal decHeight As Decimal) As Decimal
    Return (decLength * decHeight * 2) + (decWidth * decHeight * 2)
End Function

Private Function CalcArea2 (ByVal decL As Decimal, ByVal decH As Decimal) As Decimal
    Return (decL * decH)
End Function

Private Sub CalcFinalAreaAndPaint (ByRef decAreaR As Decimal, _
            ByVal decAreaW As Decimal, ByVal decAreaD As Decimal, _
            ByRef decPaintNeeded As Decimal)
    decAreaR = decAreaR – (decAreaW * 2) – decAreaD
    decPaintNeeded = decAreaR / 15
End Sub
```

QUESTION 4

4.1
```
decAmtDue = CalcAmount(intNoItems)

Private Function CalcAmount(ByVal intNo As Integer) As Decimal
    Return (intNo \ 8 * 10 + intNo Mod 8 * 1.50)
End Function
```

4.2
```
If CanYouWrite(intPredicate, intAttendance) = True Then
    lblOutcome.Text = "Congratulations, you have examination entrance"
Else
    lblOutcome.Text = "Sorry, unfortunately you don't have examination
    entrance"
End If

Private Function CanYouWrite(ByVal intPred As Integer, ByVal intAtt As Integer) _
                                                                    As Boolean
    Return (intPred >= 40 AndAlso intAtt >= 85)
End Function
```

4.3
```
intNoPages = CalcNoPages(intNoCopies, intMasterPages, blnBackToBack)

Private Function CalcNoPages(ByVal intNoC As Integer, ByVal intNoM As Integer, _
                    ByVal blnBTB As Boolean) As Integer
    Dim intPagesNeeded, intMaster As Integer
    If blnBTB = True
        intMaster = intNoM \ 2
        If (intNoM Mod 2) = 1 Then
            intMaster += 1
        End If
    Else
        intMaster = intNoM
    End If
    intPagesNeeded = intMaster * intNoC
    Return intPagesNeeded
End Function
```

QUESTION 5

Output on lblanswer1:
3.5
Output on lblAnswer2:
106
Output on lblAnswer3:
intX = 1
intY = 10
decZ = 0.00

QUESTION 6

Solve the following problems by first doing the planning and then writing the VB.NET program.

6.1 At the Cheap Unfair Store, salaries for employees are computed as follows: A male employee earns a basic salary of R2500 per month. He receives an additional R250 for every year of experience. He also receives an additional R550 if he has a qualification. Males pay 10% tax and 8% pension fund, which is deducted from the gross salary to give his net salary. Female employees earn a basic salary of R2000 per month and an additional R200 for every year of experience. Females receive an additional R450 if they have a qualification and they pay 9.5% tax and 7.5% pension fund. Enter the gender, years experience and an indication if the person has a qualification or not. Call a separate function for females and males, which must each return the gross monthly salary. Then call a procedure to calculate the tax amount and pension fund amount. Call another function to return the net salary for this person. Display the gross and net salaries in the main module. You may assume that the input values will be valid.

Planning:

	Description	Type	Name of variable
Function CalcGrossM			
Parameters	copy of years experience	integer	valYears
Input:	qualification	character	qual
Function CalcGrossF			
Parameters:	copy of years experience	integer	valYears
Input:	qualification	character	qual
Sub Procedure CalcTaxPension			
Parameters:	copy of gross salary	real	valSal
	copy of tax percentage	real	valTPerc
	copy of pension percentage	real	valPPerc
	address of tax		refTax
	address of pension		refPens
Main procedure (CalculateIncome):			
Input:	gender	character	gender
	years experience	integer	years
Output:	gross salary	real	grossSal
	tax amount	real	taxAmt
	pension amount	real	pensAmt
	net salary	real	netSal

Algorithm:

```
Function CalcGrossM(valYears)
        sal = 2500 + valYears * 250
        display "Enter qualification Y or N"
        enter qual
        if qual = "Y"
                sal = sal + 550
        endif
return sal

function CalcGrossF(valYears)
        sal = 2000 + valYears * 200
        display "Enter qualification Y or N"
        enter qual
        if qual = "Y"
                sal = sal + 450
        endif
return sal

Sub Procedure CalcTaxPension (valSal, valTPerc, valPPerc, refTax, refPens)
        RefTax = valSal * valTPerc / 100
        RefPens = valSal * valPPerc / 100
End sub

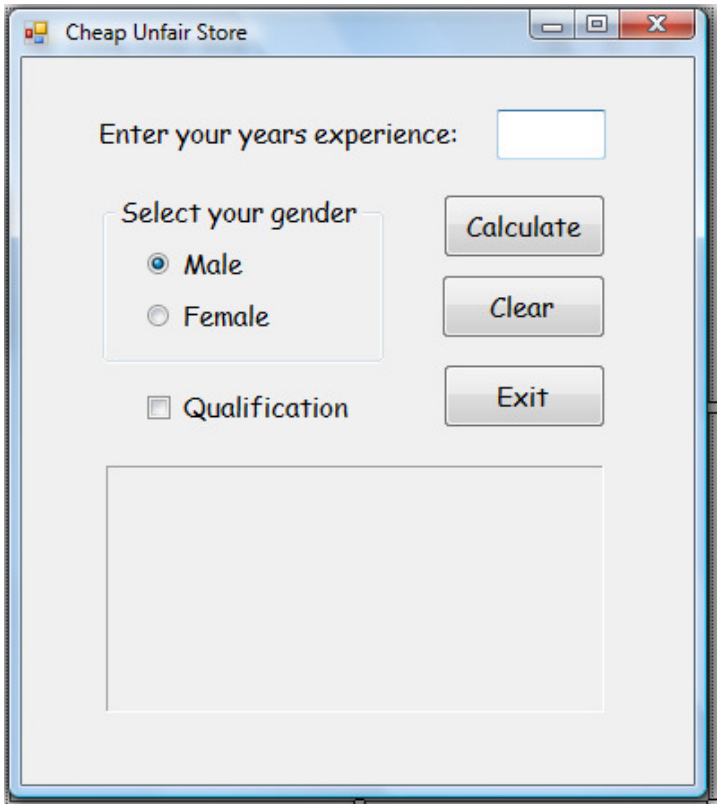
Function CalcNet (valGross, valTax, valPen)
Return (valGross – valTax – valPen)

CalculateIncome                    ~  main procedure

        display "enter gender M or F"
        enter gender
        display "enter number of years experience"
        enter years
        if gender = "M"
                grossSalary = CalcGrossM(years)
                call CalcTaxPension(grossSalary, 10, 8, taxAmt, pensAmt)
        else
                grossSalary = CalcGrossF(years)
                call CalcTaxPension(grossSalary, 9.5, 7.5, taxAmt, pensAmt)
        endif

        netSal = CalcNet(grossSalary, taxAmt, pensAmt)
        display "Your gross salary is R", grossSalary
        display "You must pay R", taxAmt, " tax"
        display "You must pay R", pensAmt, " pension fund"
        display "your net salary = R", netSal
end
```

User Interface:



Code in the code editor:

```
Public Class frmCheapUnfair

Private Function CalcGrossM(ByVal intYears As Integer) As Decimal
   Dim decSal As Decimal
   decSal = 2500 + intYears * 250
   If chkQual.Checked Then
      decSal = decSal + 550
   End If
   Return decSal
End Function

Private Function CalcGrossF(ByVal intYrs As Integer) As Decimal
   Dim decSalary As Decimal
   decSalary = 2000 + intYrs * 200
   If chkQual.Checked Then
      decSalary = decSalary + 450
   End If
   Return decSalary
End Function

Private Sub CalcTaxPension(ByVal decSal As Decimal, _
      ByVal decTPerc As Decimal, ByVal decPPerc As Decimal, _
      ByRef decTax As Decimal, ByRef decPens As Decimal)
   decTax = decSal * decTPerc / 100D
   decPens = decSal * decPPerc / 100D
End Sub
```

```
Private Function CalcNet(ByVal decGr As Decimal, ByVal decTax As Decimal, _
                         ByVal decPen As Decimal) As Decimal
   Return decGr - decTax - decPen
End Function

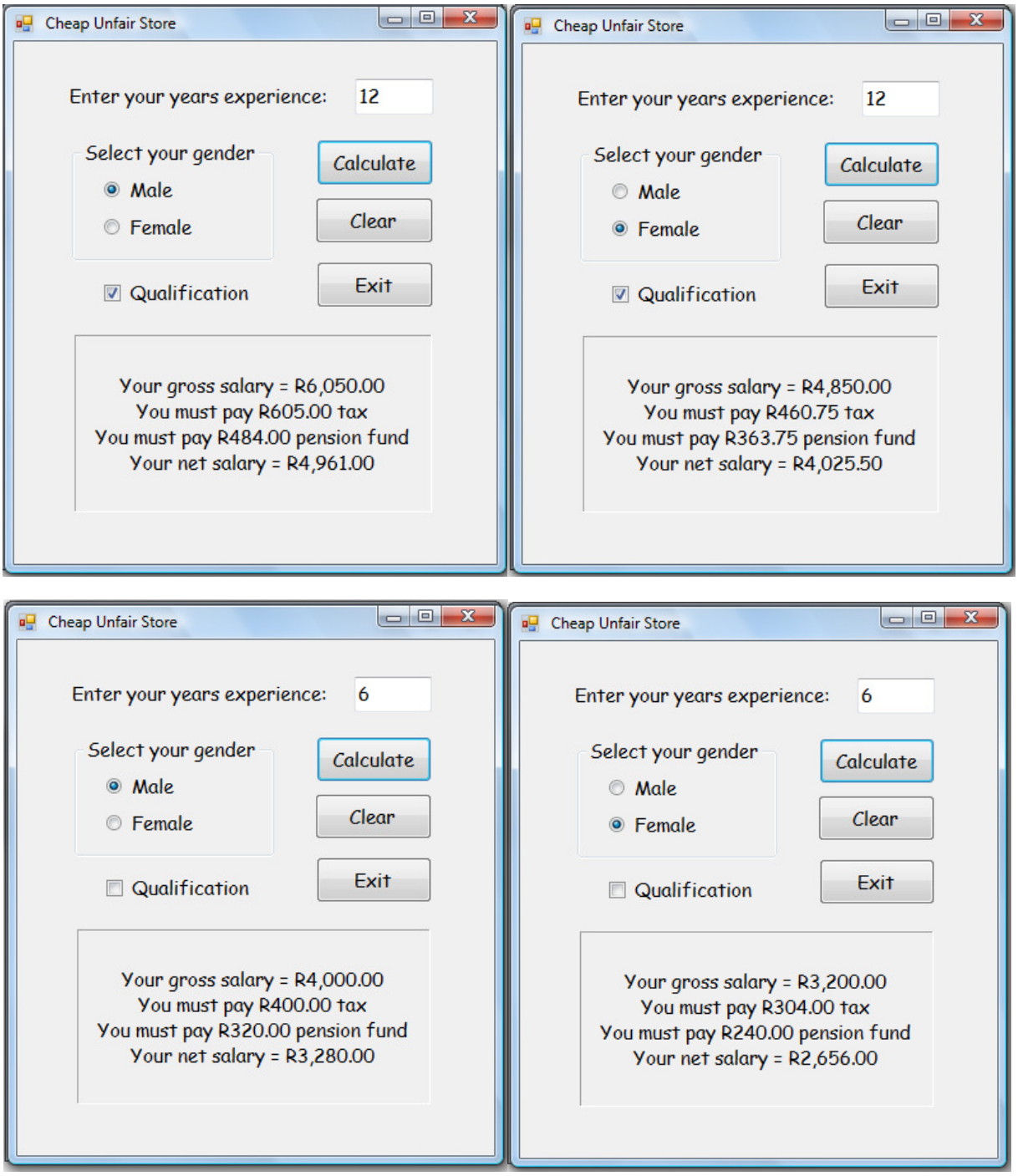
Private Sub btnCalculate_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCalculate.Click
   Dim intYearsExperience As Integer
   Dim decGrossSalary, decNetSalary, decTaxAmt, decPensAmt As Decimal
   Integer.TryParse(txtYears.Text, intYearsExperience)
   If radMale.Checked Then
      decGrossSalary = CalcGrossM(intYearsExperience)
      CalcTaxPension(decGrossSalary, 10, 8, decTaxAmt, decPensAmt)
   Else
      decGrossSalary = CalcGrossF(intYearsExperience)
      CalcTaxPension(decGrossSalary, 9.5, 7.5, decTaxAmt, decPensAmt)
   End If
   decNetSalary = CalcNet(decGrossSalary, decTaxAmt, decPensAmt)
   lblAnswer.Text = "Your gross salary = R" & _
                    decGrossSalary.ToString("N2") & _
                    ControlChars.NewLine & "You must pay R" & _
                    decTaxAmt.ToString("N2") & " tax" & _
                    ControlChars.NewLine & "You must pay R" & _
                    decPensAmt.ToString("N2") _
                    & " pension fund" & ControlChars.NewLine & _
                    "Your net salary = R" & decNetSalary.ToString("N2")
End Sub

Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnClear.Click
   txtYears.Text = ""
   lblAnswer.Text = ""
   radMale.Checked = True
   radFemale.Checked = False
   chkQual.Checked = False
   txtYears.Focus()
 End Sub

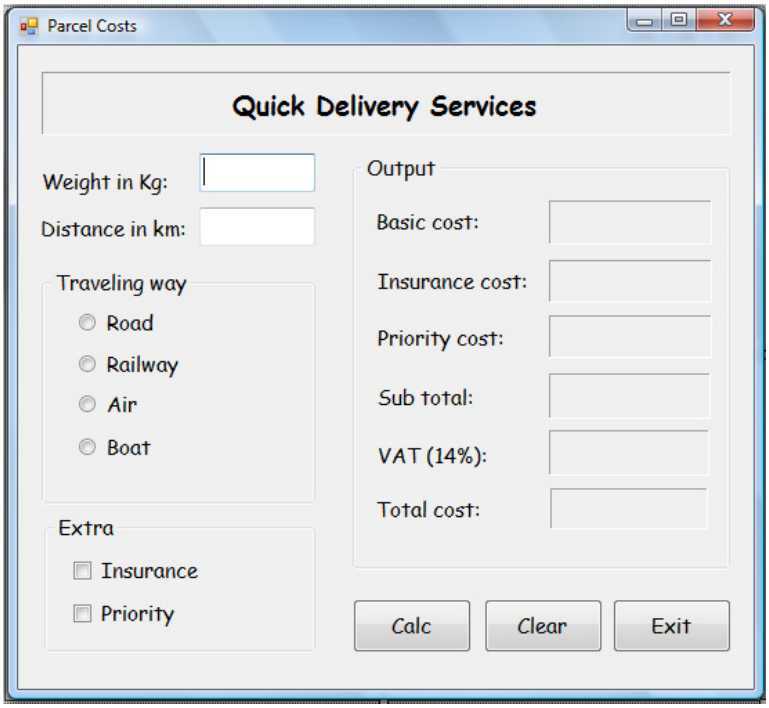
Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnExit.Click
    Close()
End Sub

End Class
```

Examples of output:



6.2 In chapter 2, in 2.5.1, the following user interface and example are used to demonstrate the call of a sub procedure without parameters when the controls are cleared.



Plan and write the calculate button for this application where parcel costs are calculated. The basic cost of a parcel is determined by the weight of the parcel, the way the parcel will be sent as well as the travelling distance. The basic cost is calculated as R1.23 per kg multiplied by the distance and the travelling cost according to the tariffs in the following table:

Road	R0.15
Railway	R0.12
Air	R0.36
Boat	R0.25

Call a function to determine the travelling cost and then call a second function to calculate and return the basic cost. Call a sub procedure to calculate the insurance cost and priority costs. If the client chooses that the parcel must be insured, the insurance cost will be 11% of the basic cost. If the client indicates that the parcel is a priority, an additional 15% will be added to the basic cost. Call a function to calculate the sub total at this stage (basic cost + insurance cost + priority cost). The main module must then calculate and add 14% VAT to the sub total to calculate the total cost of the parcel. A sub procedure must be called to display all answers.

Algorithm:

```
Function DetermineTravCost ( )

        display "enter means of transport code: "
        display  "Road – R      Railway – W"
        display "Air – A        Boat – B"
        enter trCode
        select case trCode
                case "R"
                        cost = 0.15
                case "W"
                        cost = 0.12
                case "A"
                        cost = 0.36
                case "B"
                        cost = 0.25
        end select
return cost

function CalcBasic (valKg, valKm, valCost)

return (1.23 * valKg * valKm * valCost)

Sub Procedure CalcInsurancePriorityCost (valBasic, refIns, refP)
        display "Insurance needed?  Y or N"
        enter ins
        display "Priotiry needed?  Y or N"
        enter priority
        if ins = "Y"
                refIns = valBasic * 0.11
        else
                refIns = 0
        endif
        if priority = "Y"
                refP = valBasic * 0.15
        else
                refP = 0
        endif
end sub

Function CalcSubTotal (valB, valI, valP)
return (valB + valI + valP)

Sub Procedure DisplayAnswers (valBasic, valIns, valPrio, valSub, valVat, valTot)
        display "The basic cost is R", valBasic
        display "The insurance cost is R", valIns
        display "The priority cost is R", valPrio
        display "The subtotal is R", valSub
        display "The VAT is R", valVat
        display "The total cost is R", valTot
end sub
```

```
CalculateParcelCost                          ~  main procedure
        display "enter the weight in kg"
        enter weight
        display "enter the distance in km"
        enter distance
        travelCost = DetermineTravelCost( )
        basicCost = CalcBasic (weight, distance, travelCost)
        call CalcInsurancePriorityCost (basicCost, insurance, priority)
        subtotal = CalcSubTotal (basicCost, insurance, priority)
        vat = subtotal * 0.14
        totalCost = subtotal + vat
        call DisplayAnswers (basicCost, insurance, priority, subtotal, vat, totalCost)
end
```

Code in the code editor:

```
Public Class Parcel

Private Sub ClearControls()
   txtDistance.Text = ""
   lblBasicCost.Text = ""
   lblInsuranceCost.Text = ""
   lblPriorityCost.Text = ""
   lblSubTotal.Text = ""
   lblVat.Text = ""
   lblTotalCost.Text = ""
   radRoad.Checked = True
   radRail.Checked = False
   radAir.Checked = False
   radBoat.Checked = False
   chkInsurance.Checked = False
   chkPriority.Checked = False
   txtWeight.Focus()
End Sub

Private Function DetermineTravCost() As Decimal
    Dim decCost As Decimal
    If radRoad.Checked Then
       decCost = 0.15
    ElseIf radRail.Checked Then
       decCost = 0.12
    ElseIf radAir.Checked Then
       decCost = 0.36
    ElseIf radBoat.Checked Then
       decCost = 0.25
    End If
    Return decCost
 End Function
Private Function CalcBasic(ByVal decKg As Decimal, _
        ByVal decKm As Decimal, ByVal decCost As Decimal) As Decimal
    Return (1.23 * decKg * decKm * decCost)
End Function

Private Sub CalcInsurancePriorityCost(ByVal decBasic As Decimal, _
            ByRef decIns As Decimal, ByRef DecP As Decimal)
    If chkInsurance.Checked Then
```

```
        decIns = decBasic * 0.11
    Else
        decIns = 0
    End If
    If chkPriority.Checked Then
        DecP = decBasic * 0.15
    Else
        DecP = 0
    End If
End Sub

Private Function CalcSubTotal(ByVal decB As Decimal, _
        ByVal decI As Decimal, ByVal decP As Decimal) As Decimal
    Return (decB + decI + decP)
End Function

Private Sub Displayanswers(ByVal decBasic As Decimal, _
        ByVal decIns As Decimal, ByVal decPrio As Decimal, _
        ByVal decSub As Decimal, ByVal decVat As Decimal, _
        ByVal decTot As Decimal)
   lblBasicCost.Text = "R" & decBasic.ToString("N2")
   lblInsuranceCost.Text = "R" & decIns.ToString("N2")
   lblPriorityCost.Text = "R" & decPrio.ToString("N2")
   lblSubTotal.Text = "R" & decSub.ToString("N2")
   lblVat.Text = "R" & decVat.ToString("N2")
   lblTotalCost.Text = "R" & decTot.ToString("N2")
End Sub

Private Sub btnCalc_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCalc.Click
   Dim decWeight, decDistance As Decimal
   Dim decBasicCost, decInsurance, decPriority, decSubTotal, decVat, _
       decTotalCost As Decimal
   Dim decTravelCost As Decimal
   Dim blnValid = True
   Decimal.TryParse(txtWeight.Text, decWeight)
   Decimal.TryParse(txtDistance.Text, decDistance)
   decTravelCost = DetermineTravCost()
   decBasicCost = CalcBasic(decWeight, decDistance, decTravelCost)
   Call CalcInsurancePriorityCost(decBasicCost, decInsurance, decPriority)
   decSubTotal = CalcSubTotal(decBasicCost, decInsurance, decPriority)
   decVat = decSubTotal * 0.14
   decTotalCost = decSubTotal + decVat
   Call Displayanswers(decBasicCost, decInsurance, decPriority, _
                       decSubTotal, decVat, decTotalCost)
End Sub

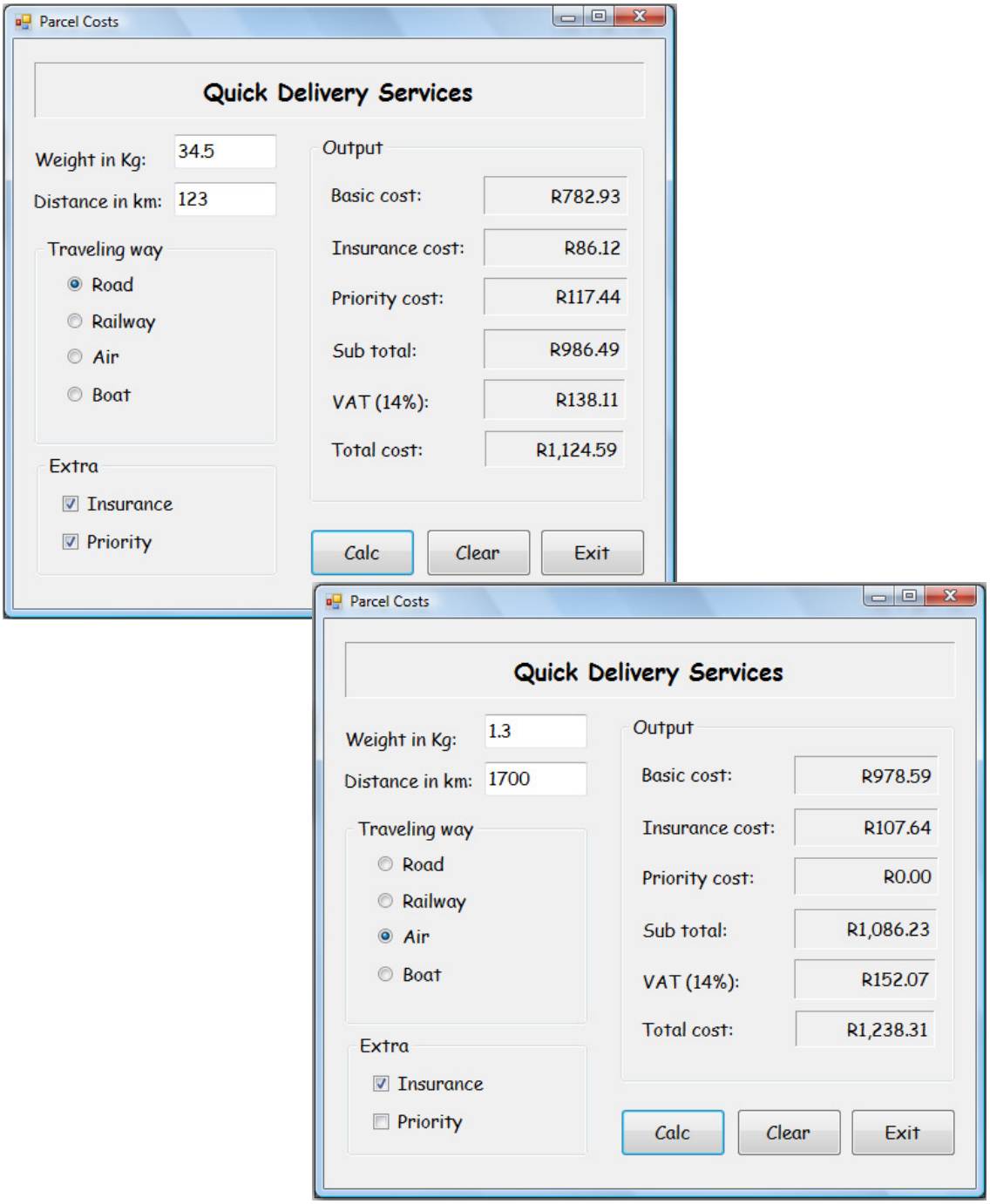
Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnClear.Click
   txtWeight.Text = ""
   Call ClearControls()
End Sub

Private Sub txtWeight_TextChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles txtWeight.TextChanged
    Call ClearControls()
End Sub
```

```
Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnExit.Click
    Close()
End Sub

End Class
```

Examples of output:



6.3 *TREAT YOURSELF* is a guesthouse with 20 rooms that can all be utilized as either a single or a double room. The tariff for a single room is R250.50 per night and the tariff for a double room is R400.90 for two persons sharing. The prices include breakfast.

Your program must produce the following information at the beginning of every week.

- Bookings for every night (number of single and number of double rooms),
- Number of people to expect for breakfast the following day,
- Income for the day

The total income for the week must also be calculated and displayed.

Display the number of single rooms, number of double rooms, number of breakfasts as well as the expected income for every day of the week. This program must provide for all possible input errors. E.g. If the sum of the number of single rooms and number of double rooms exceed 20, an appropriate error message must be displayed and the user must be able to rectify the mistake.

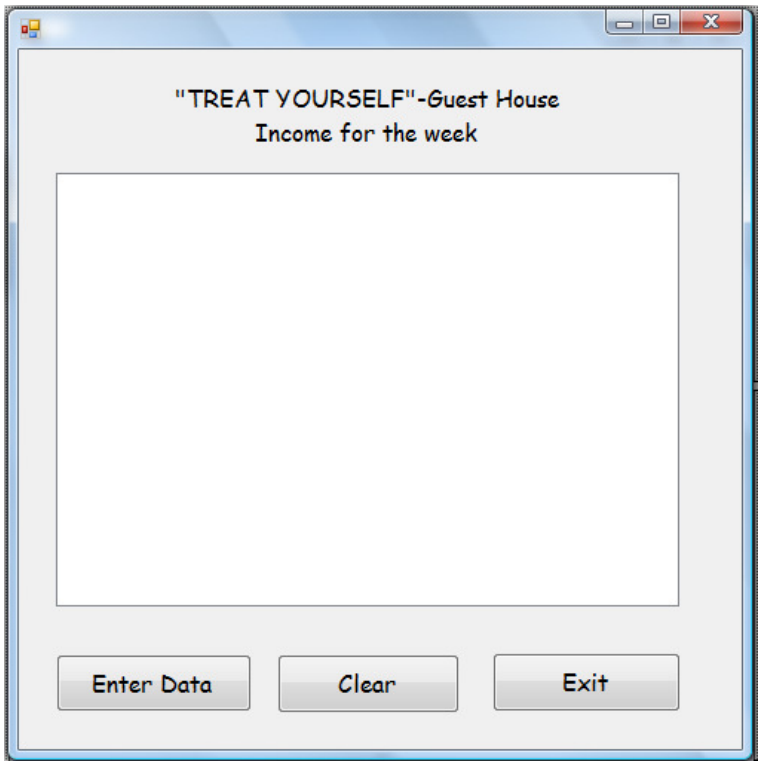
Use a list box on your user interface to display the results and make use of the PadLeft and PadRight methods in order to display the output in columns.

Provide for the following functions and independent sub procedures:

- A **sub procedure, InputData**, to enter the number of single room bookings and the number of double room bookings for the night. The total number of rooms may not be more than 20. In such a case, a suitable error message must be displayed and the user must re-enter the values until the input data is correct.
- A **function, NumberForBreakfast**, to calculate and return the number of people to expect for breakfast the following morning,
- A **function, IncomeOneNight**, to receive the number of single room bookings and the number of double room bookings and to return the income for that night,
- A **sub procedure, CalcTotalIncome**, to accumulate the total income for the week.

The information for every night (Day number, number of single room bookings, number of double room bookings, number of people to expect for breakfast and income for the night) must be displayed in the main algorithm. The total income for the week must also be displayed in the main algorithm after the loop (for-loop for 7 nights) has terminated.

User Interface:



Code in the code editor:

```
Public Class TrearYourself

Private Sub InputData(ByVal intDay As Integer, _
        ByRef intNoSingle As Integer, ByRef intNoDouble As Integer)
   Dim strSingle, strDouble As String
   Dim blnValid As Boolean = False
   Do
     strSingle = InputBox("Enter number of single rooms for day " &_
                          intDay, "Enter data")
     strDouble = InputBox("Enter number of double rooms for day " & _
                          intDay, "Enter data")
     If Integer.TryParse(strSingle, intNoSingle) AndAlso _
                         Integer.TryParse(strDouble, intNoDouble) Then
        If intNoSingle + intNoDouble > 20 Then
           MessageBox.Show("There are only 20 rooms - please re-enter", _
                "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
        Else
           blnValid = True
        End If
     Else
        MessageBox.Show("The values must be integers - please re-enter", _
                 "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
     End If
  Loop Until blnValid = True
End Sub

Private Function CalcNoBreakfast(ByVal intS As Integer, _
                                 ByVal intD As Integer) As Integer
   Return (intS + intD * 2)
End Function
```

```
Private Function IncomeOneNight(ByVal intS As Integer, _
                                ByVal intD As Integer) As Decimal
   Return (intS * 250.5 + intD * 400.9)
End Function

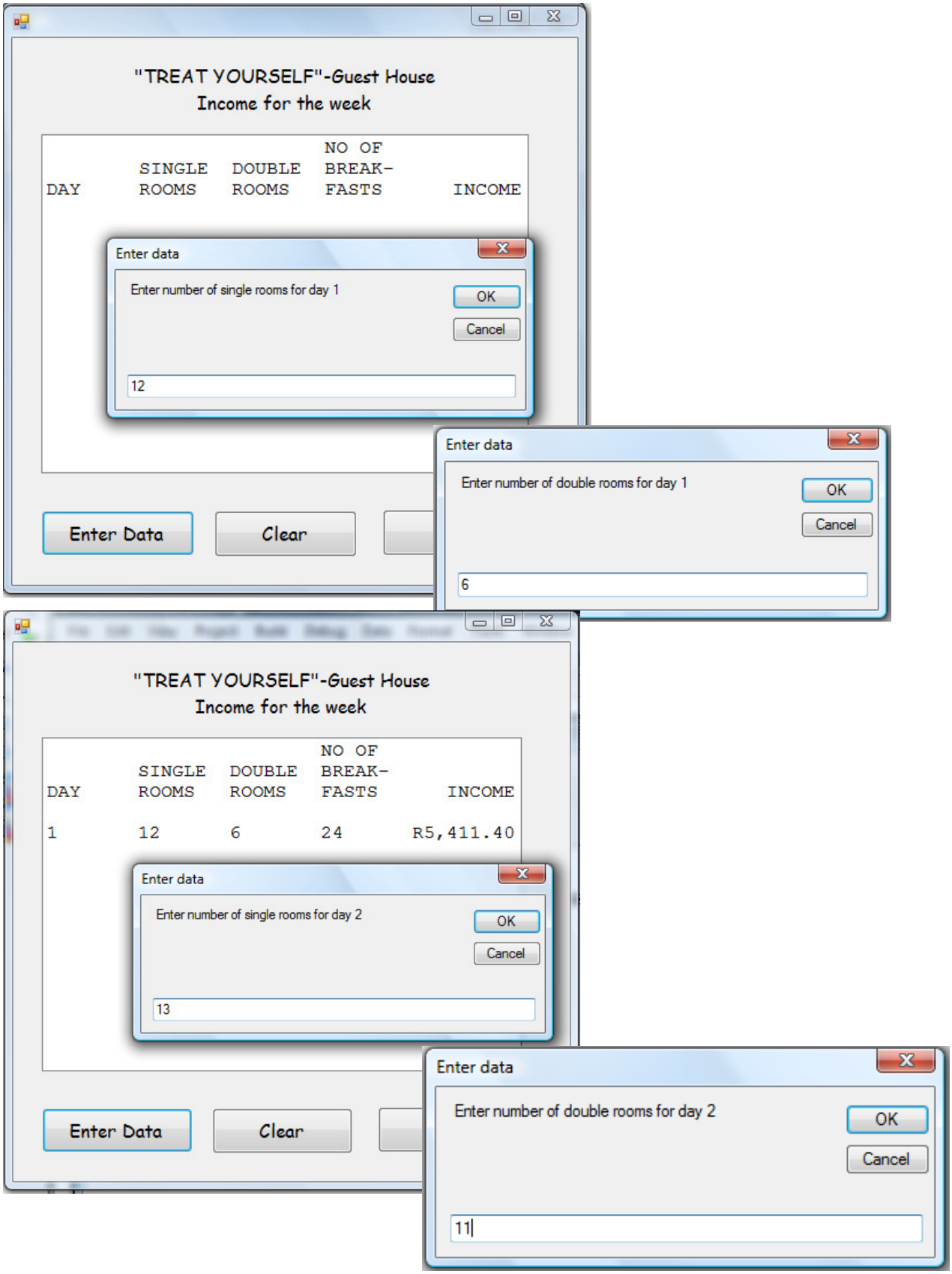
Private Sub CalcTotalIncome(ByVal decDaily As Decimal, _
                            ByRef decWeekly As Decimal)
   decWeekly += decDaily
End Sub

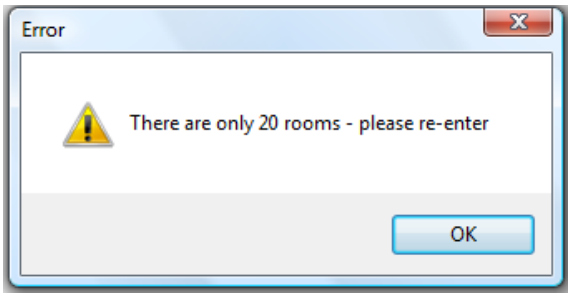
Private Sub btnEnterData_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnEnterData.Click
   Dim intNoSingle, intNoDouble, intNoPeopleBreakfast As Integer
   Dim decDailyIncome As Decimal = 0
   Dim decWeeklyIncome As Decimal = 0
   Dim intDay As Integer
   lstOutput.Items.Add("NO OF".PadLeft(29))
   lstOutput.Items.Add("SINGLE".PadLeft(14) & "DOUBLE".PadLeft(8) & _
                       "BREAK-".PadLeft(8))
   lstOutput.Items.Add("DAY".PadRight(8) & "ROOMS".PadRight(8) & _
                       "ROOMS".PadRight(8) & "FASTS".PadRight(11) & _
                       "INCOME")
   lstOutput.Items.Add(" ")
   For intDay = 1 To 7
       Call InputData(intDay, intNoSingle, intNoDouble)
       intNoPeopleBreakfast = CalcNoBreakfast(intNoSingle, intNoDouble)
       decDailyIncome = IncomeOneNight(intNoSingle, intNoDouble)
       Call CalcTotalIncome(decDailyIncome, decWeeklyIncome)
       lstOutput.Items.Add(intDay.ToString.PadRight(8) & _
                           intNoSingle.ToString.PadRight(8) & _
                           intNoDouble.ToString.PadRight(8) & _
                           intNoPeopleBreakfast.ToString.PadRight(8) & _
                           "R" & decDailyIncome.ToString("N2"))
   Next
   lstOutput.Items.Add(" ")
   lstOutput.Items.Add("TOTAL INCOME FOR THE WEEK:".PadRight(31) & _
                       "R" & decWeeklyIncome.ToString("N2"))
End Sub

Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnClear.Click
   lstOutput.Items.Clear()
End Sub

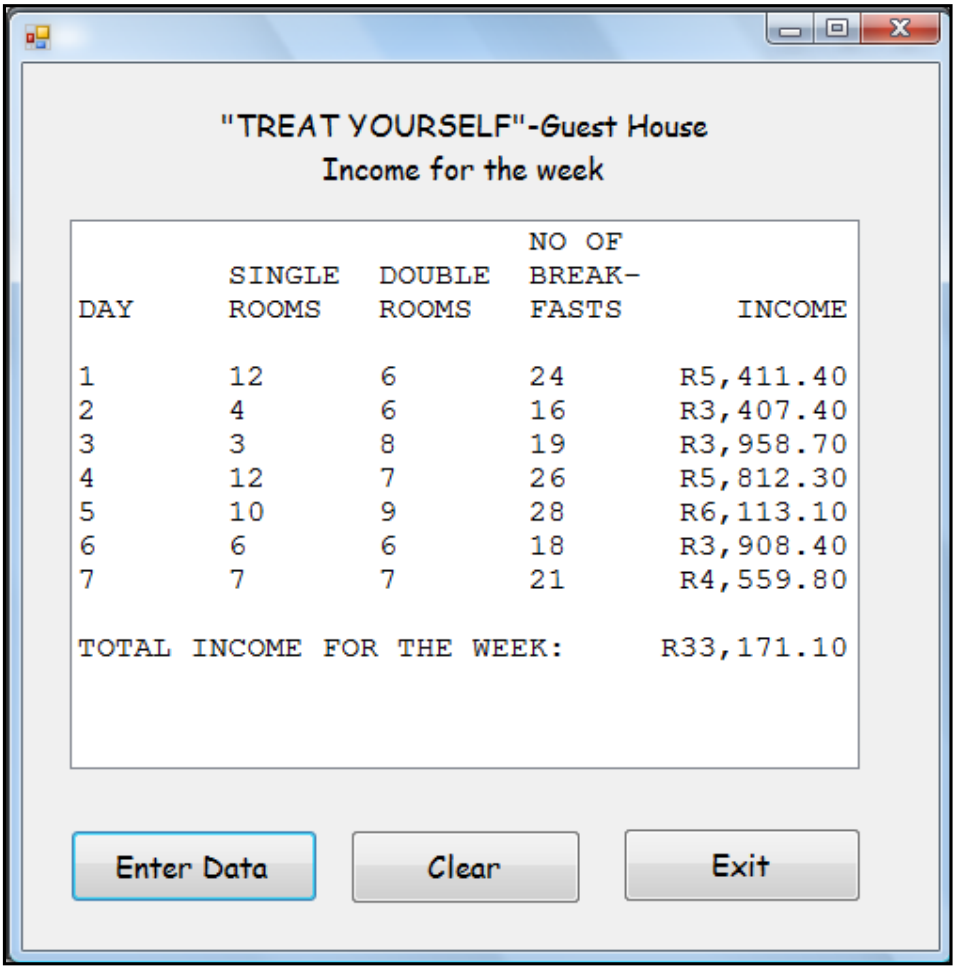
Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnExit.Click
   Close()
End Sub
End Class
```

Examples of output:





Then after it has been re-entered and valid values for the rest of the week have been entered, the output could be as follows:



6.4 TEN TRUCKS TRANSPORT is a transport company that delivers parcels to various destinations. The company has trucks with a capacity of 500kg, 750kg, 1000 kg and 1250 kg. The majority of the trucks have a capacity of 500 kg.

The cost of a parcel depends on the distance to the destination as well as the weight of the parcel and customers pay R1 per km as well as R1.23 per kg of the parcel weight.

Before a truck departs, the distance to the destination must be entered as an integer value. The capacity of the truck must also be selected. The weights of all the parcels must then be entered and after every parcel weight has been entered, the amount payable must be calculated and the parcel number, weight and amount payable must be displayed.

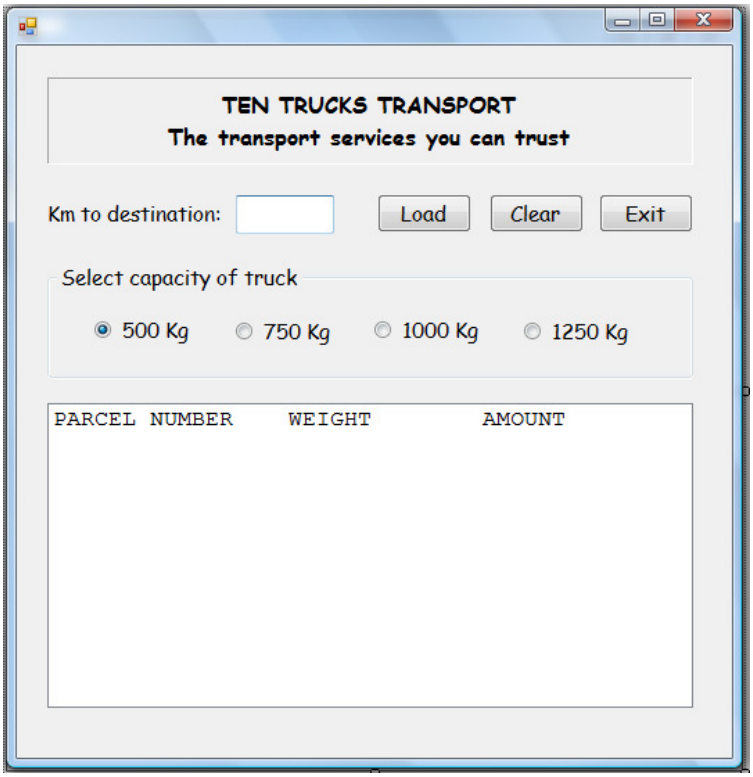
Parcels may be added as long as the current load plus the parcel weight is within the capacity of the truck. The truck need not be full on departure and the entering process can therefore be terminated before the truck is full. It is also possible that one parcel could be too heavy and will cause that the capacity of the truck is exceeded, in which case a parcel of acceptable weight may replace it.

When the weight of all the parcels has been entered, the total amount due for the load must be displayed. A message indicating whether the truck is completely full or not is also displayed. If the truck is not completely full, the available space (in kg) must be displayed.

The program must provide for all possible input errors, in which case a suitable error message must be displayed.

Call a function to determine the capacity of the truck and call a sub procedure to handle all calculations for a valid parcel to be loaded onto the truck.

User Interface:



Because the majority of the trucks have a capacity of 500kg, it will be set as the default radio button when the form is loaded and when the clear button is clicked. The headings in the list box will also be displayed when the form is loaded. The PadLeft and PadRight methods will be used to display output in columns in the list box.

Code in the code editor:

```
Public Class TenTrucks

Private Function DetermineCapacity() As Decimal
   Dim decCap As Decimal
   If rad500.Checked Then
      decCap = 500D
   ElseIf rad750.Checked Then
      decCap = 750D
   ElseIf rad1000.Checked Then
      decCap = 1000D
   Else
      decCap = 1250D
   End If
   Return decCap
End Function

Private Sub LoadValidParcel(ByVal intDistance As Integer, _
        ByVal decWeight As Decimal, ByRef decAmt As Decimal, _
        ByRef decTotalKg As Decimal, ByRef decTotAmt As Decimal)
   decTotalKg += decWeight
   decAmt = intDistance + 1.23 * decWeight
   decTotAmt += decAmt
End Sub

Private Sub btnLoad_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnLoad.Click
   Dim strKg As String
   Dim intKmDestination, intCounter As Integer
   Dim decAmount, decTotalAmt, decAccKg, decCapacity, decKg As Decimal
   decCapacity = DetermineCapacity()
   If Integer.TryParse(txtKm.Text, intKmDestination) AndAlso _
                       intKmDestination > 0 Then
      decTotalAmt = 0
      decAccKg = 0
      strKg = InputBox("Enter the weight of the first parcel", _
                       "Enter in kg")
      intCounter = 1
      Do
         If Decimal.TryParse(strKg, decKg) And decKg > 0 Then
            If (decAccKg + decKg) <= decCapacity Then
               Call LoadValidParcel(intKmDestination, decKg, _
                                    decAmount, decAccKg, decTotalAmt)
               lstOutput.Items.Add(intCounter.ToString.PadLeft(7) & _
                         decKg.ToString("N2").PadLeft(14) & "kg" & _
                         decAmount.ToString("N2").PadLeft(14))
               intCounter += 1
               strKg = InputBox("Enter the weight of the next parcel", _
                                "Enter in kg")
            Else
               MessageBox.Show("Not enough space for this parcel", _
                               "You can try a smaller parcel", _
                               MessageBoxButtons.OK, _
                               MessageBoxIcon.Exclamation)
```

```
                strKg = InputBox("Enter the weight of the next parcel", _
                                 "Enter in kg")
            End If
         Else
            MessageBox.Show("The weight in kg must be a + decimal value", _
                            "Please re-try", MessageBoxButtons.OK, _
                            MessageBoxIcon.Exclamation)
            strKg = InputBox("Enter the weight of the next parcel", _
                             "Enter in kg")
         End If
      Loop Until decAccKg > decCapacity OrElse strKg = ""
      lstOutput.Items.Add(" ")
      lstOutput.Items.Add(decTotalAmt.ToString("N2").PadLeft(37))
      lstOutput.Items.Add(" ")
      If decAccKg < decCapacity Then
         lstOutput.Items.Add("The truck is not full - " & _
                 decCapacity - decAccKg).ToString("N2") & "kg space left")
      Else
         lstOutput.Items.Add("The truck is full")
      End If
   Else
      MessageBox.Show("The km must be an integer and > 0", _
                      "Please re-enter", MessageBoxButtons.OK, _
                      MessageBoxIcon.Exclamation)
      txtKm.Text = ""
      txtKm.Focus()
   End If
End Sub

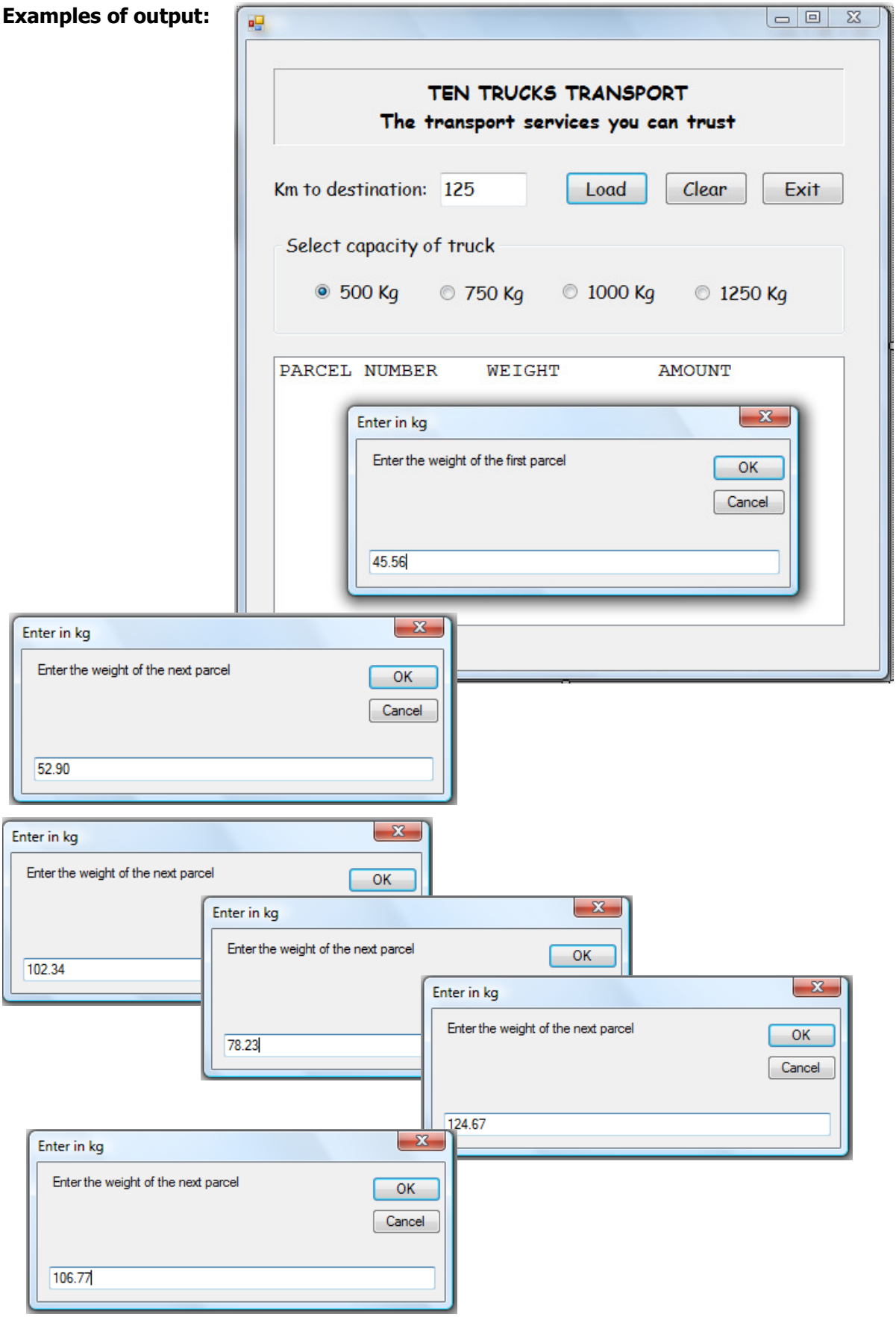
Private Sub Prac6_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
   rad500.Checked = True
   lstOutput.Items.Add("PARCEL NUMBER" & "WEIGHT".PadLeft(10) & _
                       "AMOUNT".PadLeft(14))
   lstOutput.Items.Add(" ")
End Sub

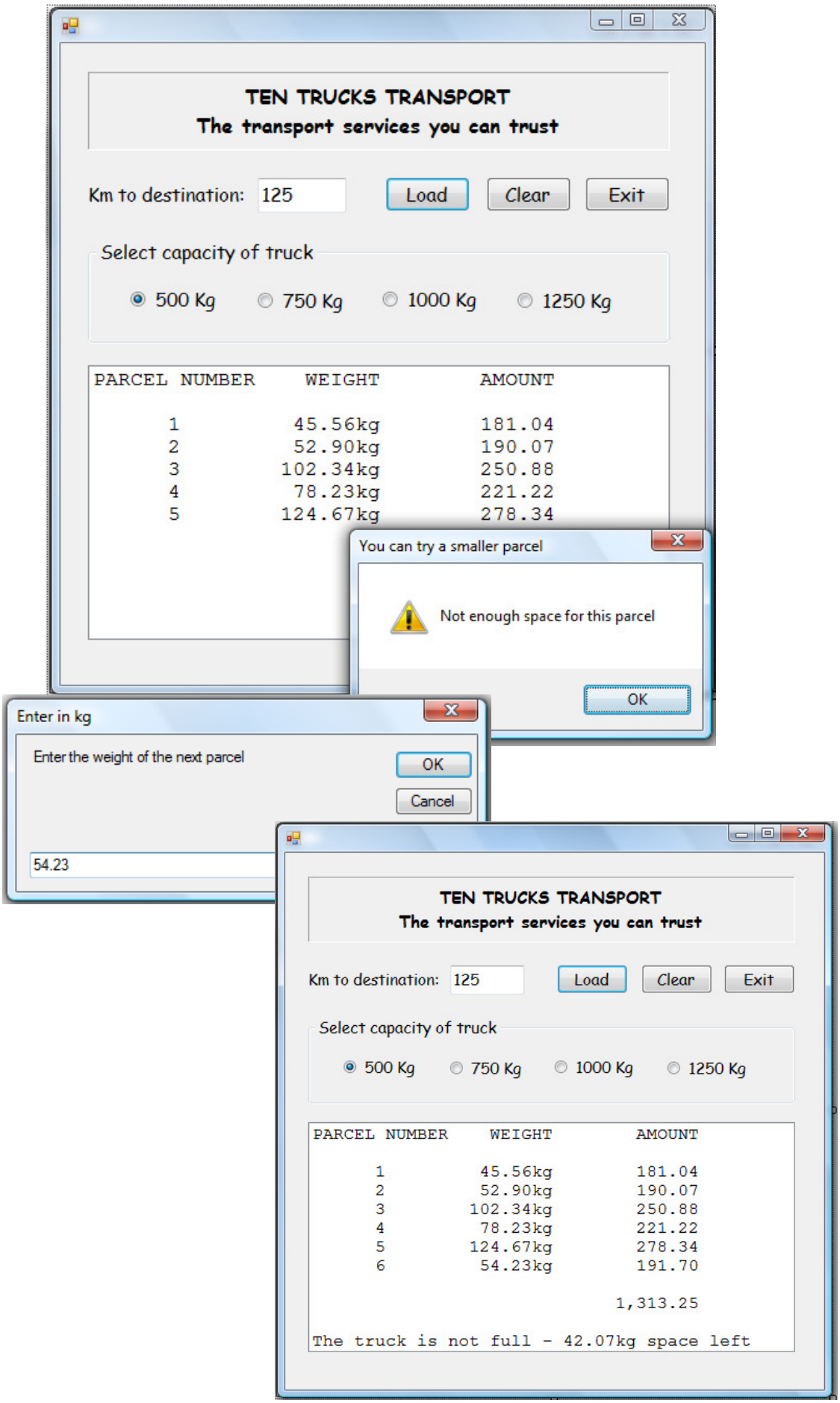
Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnClear.Click
   txtKm.Text = ""
   txtKm.Focus()
   lstOutput.Items.Clear()
   rad500.Checked = True
   lstOutput.Items.Add("PARCEL NUMBER" & "WEIGHT".PadLeft(10) & _
                       "AMOUNT".PadLeft(14))
End Sub

Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnExit.Click
   Close()
End Sub

End Class
```

Examples of output:




TEN TRUCKS TRANSPORT
The transport services you can trust
Km to destination: 125
Load
Clear
Exit
Select capacity of truck
500 Kg
750 Kg
1000 Kg
1250 Kg
PARCEL NUMBER WEIGHT AMOUNT
1 45.56kg 181.04
2 52.90kg 190.07
3 102.34kg 250.88
4 78.23kg 221.22
5 124.67kg 278.34
You can try a smaller parcel
Not enough space for this parcel
OK
Enter in kg
Enter the weight of the next parcel
OK
Cancel
54.23
TEN TRUCKS TRANSPORT
The transport services you can trust
Km to destination: 125
Load
Clear
Exit
Select capacity of truck
500 Kg
750 Kg
1000 Kg
1250 Kg
PARCEL NUMBER WEIGHT AMOUNT
1 45.56kg 181.04
2 52.90kg 190.07
3 102.34kg 250.88
4 78.23kg 221.22
5 124.67kg 278.34
6 54.23kg 191.70
1,313.25
The truck is not full - 42.07kg space left

CHAPTER 3
QUESTIONS & ANSWERS
ARRAYS

QUESTIONS:

QUESTION 1

Indicate whether the following statements are TRUE or FALSE. Provide a reason for your answer.

1.1 The index of the last element of an array is the number of elements in the array minus 1.
1.2 The elements of the array called numberArray may contain both integers and numbers of the type double.
1.3 Parallel arrays have the same number of elements.
1.4 When referring to a row in a two-dimensional array, two indexes must be used.
1.5 Once initial values are assigned to elements of an array, the contents cannot be changed.
1.6 It is possible to access an individual element in an array.
1.7 It is possible to declare an array that may contain string values only.
1.8 A two-dimensional array of 3 rows and 5 columns cannot serve the same purpose as 3 parallel arrays of 5 elements each.
1.9 When loading a two-dimensional array with data, the values are always loaded row by row, not column by column.
1.10 The first subscript for a two-dimensional array refers to the column and the second subscript refers to the row.
1.11 The following declaration of a two-dimensional array is valid:
```
Dim stMarks(6) As Integer
```
1.12 A subscript may be called an index.
1.13 The Array.Sort method can be used to sort the values of two parallel arrays into ascending order.
1.14 The For Each . . Next statement can be used to store values in an array.
1.15 The following statement is valid:
```
decNumbers(5.5) = 5.5D
```

QUESTION 2

In each of the following cases, encircle the letter of the most correct answer.

2.1 An array called gArray has 20 elements and uses the variable x as the index. Which of the following statements will process the loop instructions for each element in the array?

a. do while (x < 20) where the initial value of x = 1
b. do while (x < 20) where the initial value of x = 0
c. do while (x <= 20) where the initial value of x = 0
d. do while (x < 21) where the initial value of x = 0
e. None of the above

2.2 An array of five rows and four column has _____ elements

a. 5
b. 4
c. 20
d. 16
e. 19

2.3 Paired arrays _____

a. may contain different types of data
b. must have the same number of elements
c. must be related to one-another
d. all of the above
e. None of the above

2.4 Two-dimensional arrays must _____

a. use two indexes to access a single element
b. have the same number of rows and columns
c. contain only integer values
d. Contain only decimal values
e. none of the above

2.5 The number of elements in a two-dimensional array _____

a. is equal to the number of rows multiplied by the number of columns
b. may never be more than 1024
c. is equal to the number of rows plus the number of columns
d. is equal to the number of the elements of 2 parallel arrays
e. is equal to the number of rows divided by the number of columns

2.6 An array may not _____

a. contain data before the processing of the program starts.
b. be accessed before data is loaded into the elements.
c. contain more elements than the number of rows
d. be sorted
e. none of the above

2.7 The following is true of an array:

a. it may contain more rows than columns
b. it may be loaded by using a loop structure and input from the keyboard
c. the value of the index is always an integer
d. may be sorted
e. all of the above

2.8 The following is not true of an array:

a. it may have any number of elements
b. every element is addressed by using the name of the array and an index
c. the index is always of the type integer and may have a positive or negative number
d. it can been seen as a row of elements
e. the array must be declared in the program

2.9 Three parallel arrays _____

a. are always of the same data type
b. may contain 7, 8 and 9 elements respectively
c. are usually used in the same program
d. may use different index values to refer to related elements
e. none of the above

2.10 Which of the following statements assigns the Boolean value True to the element located in the third row, first column of a two-dimensional Boolean array named `blnMedicineReceived`

a. blnMedicineReceived(0, 2) = True
b. blnMedicineReceived(1, 3) = True
c. blnMedicineReceived(3, 1) = True
d. blnMedicineReceived(2, 0) = True
e. None of the above

2.11 How many elements are in the following array?

```
Dim decTaxPerc(3, 4) As Decimal
```

a. 3
b. 4
c. 7
d. 12
e. 11

2.12 Which of the following statements declares a one-dimensional array called **`decSales`** with 7 elements?

a. Dim decSales() As Decimal
b. Dim decSales(6) As Decimal
c. Dim decSales(7) As Decimal
d. Dim decSales() As Decimal = {1000D, 2000D, 3000D, 4000D, 5000D, _
6000D, 5555D}
e. Both b and d

2.13 If an array called **`decSales`** contains 12 elements, which of the following statements can verify that the subscript **`x`** contains a valid value?

a. If decSales(x) >= 0 And decSales(x) <= 12 Then
b. If decSAles(x) >= 0 AndAlso decSales(x) <= 11 Then
c. If x >= 0 AndAlso x <= 11 Then
d. If x > 0 AndAlso x < 12 Then
e. If x >= 0 AndAlso x < 11 Then

2.14 Which of the following statements will correctly add 100 to each sale in the array called **`decSales`**? The array contains 12 elements.

a.
```
For x As Integer = 0 to 11
    decSales(x) += 100
Next
```
b.
```
For x As Integer = 0 to 11
    decSales(x) = decSales(x) + 100
Next
```
c.
```
For Each decAmt As Decimal in decSales
    decAmt = decAmt + 100
Next
```
d.
```
Dim x As Integer
Do while x <= 11
    decSales(x) += 100
    x += 1
Next
```
e. All of the above

2.15 Which of the following statements will sort the array called **`decSales`** in ascending order?

a. Sort(decSales)
b. decSales.Sort
c. SortArray(decSales)
d. Array.Sort(decSales)
e. None of the above

QUESTION 3

The array called *number* is given containing the following values:

number

1	2	3	4	5	6	7	8

Process the code given below and show the new values of the array:

```
for i = 0 to 5 step 2
        number(i) = number(i+1)
        number(i+3) = i + number(i)
next
```

number

QUESTION 4

The array *priceVal* contains 10 integer value elements. Process the code given below and show the new values of the array:

priceVal


```
for i = 0 to 9
        priceVal(i) = i * 2
next

m = 4
do while priceVal(m) > 5
        priceVal(m) = priceVal(m + 1)
        m = m – 1
loop

for j = m to 9
        priceVal(j) = priceVal(j) * 3
next
```

priceVal

QUESTION 5

The array *XYZ* contains 10 numeric elements. Process the code given below and show the new values of the array:

XYZ


```
k = 4
for x = 1 to 9
        XYZ(x) = k / 2
        k= k + 4
next

for y = 1 to 9
        XYZ(y – 1) = XYZ(y)
next

w = 0
for z = 0 to 7 step 3
        XYZ(w) = XYZ(z + 1) + 3
        w = w + 1
next
```

XYZ

QUESTION 6

Do the necessary planning and write a VB.Net solution for the following:

a. Solomon sells 40 different products that he has numbered from 1 to 40 in his fast food store. He decided to use a program to handle his sales. But first a short program must be developed to load the prices into the array in the program.

b. Add the following code to the program to calculate the amount due by a client. The client orders items by using the product number and also announcing how many of this product the client wants. The client may also order a second, a third, etc. products. For every product, the product number, number ordered, unit price and amount for these products must be displayed. The amount due by the client is displayed once the product code = 99 is entered.

QUESTION 7

Three parallel arrays contain information regarding 8 girls. The first array contains the first names of the girls, the second array contains the surnames of the girls and the last array contains the birthday (format DDMM) of each girl. Plan and develop a program to display the first name, last name and birthday in the following format:

Jenny Nkosi, 17 May

You may assume that all 3 arrays already contain values.

QUESTION 8

Two parallel arrays store the following information for 7 employees.

The first array is a one-dimensional array that contains the 7 employee numbers. The second array is a two-dimensional array that contains the basic monthly salary in the first column, the bonus in the second column and the net salary in the third column. Therefore every row in the 2-dim array, will contain the basic monthly salary, bonus and net salary for the employee whose number is stored in the parallel array in the same position.

	Emp no	B Month Sal	Bonus	Net salary
Emp 1	1234	5000.00		
Emp 2	2345	6000.00		
Emp 3	3456	6500.00		
Emp 4	4567	7000.00		
Emp 5	5678	8650.00		
Emp 6	6789	9800.00		
Emp 7	7890	9950.00		

Diagram of arrays

The employee numbers are available in the first array. The basic monthly salaries are available in the second array. Do the planning and develop a program to complete the rest of the array by using the following:

Bonus = 75% of the basic monthly salary
Deductions from basic monthly salary:

Tax = 18%
Medical aid = R425

Display all the values on the screen.

QUESTIONS:

QUESTION 1

1.1 TRUE
1.2 FALSE
All the elements in an array must contain the same type of data.
1.3 TRUE
1.4 FALSE
When referring to a row in an array, only one index is used, but when referring to an element, two indexes must be used.
1.5 FALSE
The values of the elements may be changed by processing.
1.6 TRUE
1.7 TRUE
1.8 FALSE
It can serve the same purpose.
1.9 FALSE
The array may be populated column by column using the loop that manipulates the column array as the outer loop.
1.10 FALSE
The first subscript in a 2-dim array refers to the row and the second subscript refers to the column.
1.11 FALSE
The declaration of a 2-dim array needs two subscripts
1.12 TRUE
1.13 FALSE
The Array.Sort method can only be used to sort one array in ascending sequence.
1.14 FALSE
The For Each . . Next statement can only be used to retrieve existing values into an array. These values may then be changed if necessary. However, the statement cannot be used to assign initial values to elements in an array.
1.15 FALSE
A subscript may not contain a decimal point.

QUESTION 2

2.1 b
2.2 c
2.3 d
2.4 a
2.5 a
2.6 e
2.7 e
2.8 c
2.9 c
2.10 d
2.11 d
2.12 e
2.13 c
2.14 e
2.15 d

QUESTION 3

number

2	2	2	2	4	4	7	8

QUESTION 4

Intermediate arrays are used to solve this problem:

After the first loop:
priceVal

0	2	4	6	8	10	12	14	16	18

After the second loop:
priceVal

0	2	4	10	10	10	12	14	16	18

Final answer - After the third loop:
priceVal

0	2	12	30	30	30	36	42	48	54

QUESTION 5

Intermediate arrays are used to solve this problem:

After the first loop:
XYZ

	2	4	6	8	10	12	14	16	18

After the second loop:
priceVal

2	4	6	8	10	12	14	16	18	18

Final answer - After the third loop:
priceVal

7	13	19	8	10	12	14	16	18	18

QUESTION 6

Do the necessary planning and write a VB.Net solution for the following:

a. Solomon sells 40 different products that he has numbered from 1 to 40 in his fast food store. He decided to use a program to handle his sales. But first a short program must be developed to load the prices into the array in the program.

	Description	Type	Name of variable / Array
Input:	price	real number	priceArray
Output:	prices in array	real	priceArray

Algorithm:

Solomons

```
        Declare the array priceArray with 40 elements of the type real
        ~       enter values into the array
        for w = 0 to 39
                display "Provide price for product number ", w+1
                enter priceArray(w)
        next
end
```

b. Add the following code to the program to calculate the amount due by a client. The client orders items by using the product number and also announcing how many of this product the client wants. The client may also order a second, a third, etc. products. For every product, the product number, number ordered, unit price and amount for these products must be displayed. The amount due by the client is displayed once the product code = 99 is entered.

	Description	Type	Name of variable
Input:	product number	integer	prodNo
	Number of items	integer	noItems
Output:	amount due	real	amtDue

Algorithm:

Solomons

```
        Declare the array priceArray with 40 elements of the type real
        ~       enter values into the array
        for w = 0 to 39
                display "Provide price for product number ", w+1
                enter priceArray(w)
        next
        amtDue = 0
        display "Provide the first product number of the order, 99 to terminate"
        enter prodNo
        Display headings
        do while prodNo NOT = 99
                display "Provide number of items wanted"
                enter noItems
                amount = noItems * priceArray(prodNo-1)
                amtDue = amtDue + amount
                display ProdNo, NoItems, PriceArra(ProdNo-1), amount
                display "Provide the product number, 99 to terminate"
                enter prodNo
        loop
        display "The amount due is R", amtDue
end
```

Code in the code editor:

Take note:

The array is declared as global because values are entered into the array during the form load event – otherwise, the array will have to be loaded before every next order.

No data validation is done and this program will assume that every value entered will be valid. A good exercise will be to rewrite it in order to include validation.

Values in the input box statements are too long to display in one line and cannot be broken down into two lines by means of the line continuation character. In the program it is displayed on one line.

```
Public Class Solomons

    Private decPriceArray(39) As Decimal

Private Sub btnOrder_Click(. . .) Handles btnOrder.Clic
    Dim decAmtDue As Decimal = 0
    Dim decAmount As Decimal
    Dim intProdNo, intNoItems As Integer
    Dim strProdNo, strNoItems As String
    lstOrder.Items.Add("PROD NO".PadRight(9) & _
             "NO BOUGHT".PadRight(12) & "UNIT PRICE" & _
             "AMOUNT".PadLeft(10))
    lstOrder.Items.Add(" ")
    strProdNo = InputBox("Provide the first product number of the
                order, 99 to terminate", " ")
    Do While strProdNo <> "99"
       intProdNo = Convert.ToInt32(strProdNo)
       strNoItems = InputBox("Provide number of items wanted", " ")
       intNoItems = Convert.ToInt32(strNoItems)
       decAmount = intNoItems * decPriceArray(intProdNo - 1)
       decAmtDue += decAmount
       lstOrder.Items.Add(intProdNo.ToString.PadRight(12) & _
          intNoItems.ToString.PadRight(9) & _
          decPriceArray(intProdNo - 1).ToString("N2").PadLeft(10) & _
          decAmount.ToString("N2").PadLeft(10))
       strProdNo = InputBox("Provide the next product number of the
                   order, 99 to terminate", " ")
    Loop
    lstOrder.Items.Add(" ")
    lstOrder.Items.Add(("Amount due for order = R" & _
                        decAmtDue.ToString("N2")).PadLeft(41))
End Sub

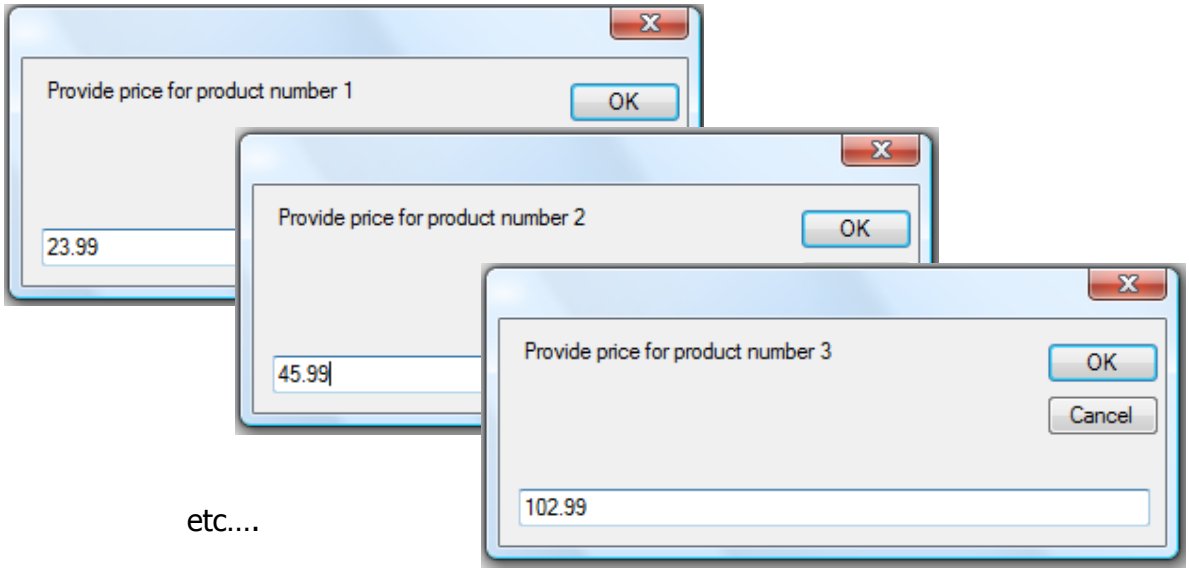
Private Sub btnNext_Click(. . .) Handles btnNext.Click
   lstOrder.Items.Clear()
End Sub

Private Sub Solomons_Load(. . .) Handles MyBase.Load
   Dim strPrice As String
   For w As Integer = 0 To 39
       strPrice = InputBox("Provide price for product number " & _
                           w + 1, " ")
            decPriceArray(w) = Convert.ToDecimal(strPrice)
   Next
End Sub

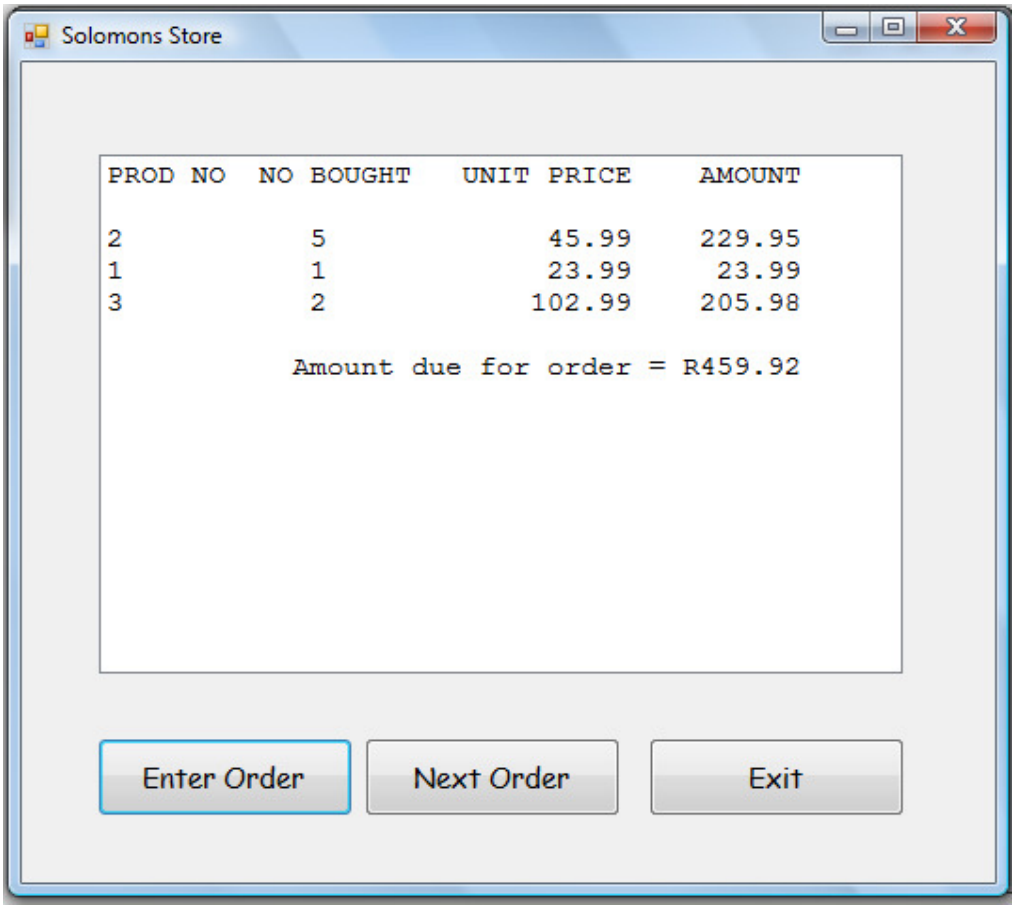
Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub

End Class
```

Example of Output:



etc....



QUESTION 7

Three parallel arrays contain information regarding 8 girls. The first array contains the first names of the girls, the second array contains the surnames of the girls and the last array contains the birthday (format DDMM) of each girl. Plan and develop a program to display the first name, last name and birthday in the following format:

Jenny Nkosi, 17 May

You may assume that all 3 arrays already contain values.

Variables used:

firstNameArray contains first names of the type string
lastNameArray contains last names of the type string
birthdayArray contains birthdays of the type integer: DDMM
monthNameArray contains the names of the months: January – December
birthDay contains the day of the birthday
birthMonth contains the month of the birthday

Algorithm:

DisplayNameAndBirtday

Declare the array firstNameArray with 8 elements of the type string
Declare the array lastNameArray with 8 elements of the type string
Declare the array birthDayArray with 8 elements of the type integer
~ declare an array that contains the names of the months of the year
~ from January to December
Declare the array monthNameArray with 12 elements of the type string

for k = 0 to 7
    birthMonth = birthdayArray(k).substring(2,2)
    birthDay = birthDayArray(k).substring(0,2)
    display firstNameArray(k), " ", lastNameArray(k), ", ",
                birthDay, " ", monthNameArray(birthMonth)
next
end

Code in the code editor:

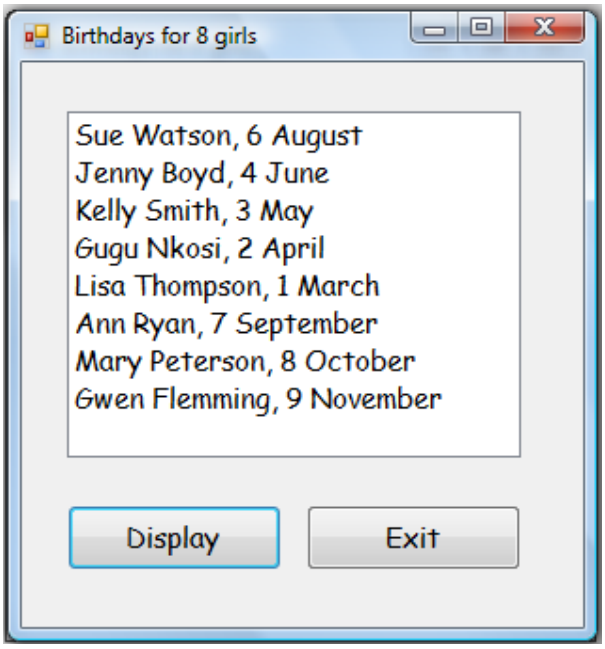
```
Public Class NameBirthDay

Private Sub btnDisplay_Click(. . .) Handles btnDisplay.Click
   Dim strFirstNameArray() As String = {"Sue", "Jenny", "Kelly", _
                  "Gugu", "Lisa", "Ann", "Mary", "Gwen"}
   Dim strLastNameArray() As String = {"Watson", "Boyd", "Smith", _
              "Nkosi", "Thompson", "Ryan", "Peterson", "Flemming"}
   Dim strBirthDate() As String = {"0607", "0405", "0304", "0203", _
                        "0102", "0708", "0809", "0910"}
   Dim strMonthNameArray() As String = {"January", "February", _
                             "March", "April", "May", "June", _
                             "July", "August", "September", _
                             "October", "November", "December"}
   Dim intBirthDay As Integer
   Dim intBirthMonth As Integer
```

```
   For k As Integer = 0 To 7
       intBirthDay = Convert.ToInt32(strBirthDate(k).Substring(0, 2))
       intBirthMonth = _
       Convert.toInt32(strBirthDate(k).Substring(2, 2))
       lstDisplay.Items.Add(strFirstNameArray(k) & " " & _
                    strLastNameArray(k) & ", " & _
                    intBirthDay & " " & _
                    strMonthNameArray(intBirthMonth))
    Next
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Output:



QUESTION 8

Two parallel arrays store the following information for 7 employees.
The first array is a one-dimensional array that contains the 7 employee numbers. The second array is a two-dimensional array that contains the basic monthly salary in the first column, the bonus in the second column and the net salary in the third column. Therefore every row in the 2-dim array will contain the basic monthly salary, bonus and net salary for the employee whose number is stored in the parallel array in the same position.

	Emp no
Emp 1	1234
Emp 2	2345
Emp 3	3456
Emp 4	4567
Emp 5	5678
Emp 6	6789
Emp 7	7890

B Month Sal	Bonus	Net salary
5000.00		
6000.00		
6500.00		
7000.00		
8650.00		
9800.00		
9950.00		

The employee numbers are available in the first array. The basic monthly salaries are available in the second array. Do the planning and develop a program to complete the rest of the array by using the following:

Bonus = 75% of the basic monthly salary
Deductions from basic monthly salary:

Tax = 18%
Medical aid = R425

Display all the values on the screen.

Code in the code editor:

```
Public Class Employee

Private Sub btnDisplay_Click(. . .) Handles btnDisplay.Click
   Dim strEmpNo() As String = {"1234", "2345", "3456", "4567", _
                               "5678", "6789", "7890"}
   Dim decSalAmounts(6, 2) As Decimal
   Dim decTaxAmt As Decimal
   Dim x As Integer

   'Assign initial values to first column of 2-dim array
   decSalAmounts(0, 0) = 5000D
   decSalAmounts(1, 0) = 6000D
   decSalAmounts(2, 0) = 6500D
   decSalAmounts(3, 0) = 7000D
   decSalAmounts(4, 0) = 8650D
   decSalAmounts(5, 0) = 9800D
   decSalAmounts(6, 0) = 9950D

   'Calculate bonusses
   For x = 0 To 6
       decSalAmounts(x, 1) = decSalAmounts(x, 0) * 0.75
   Next

   'Calculate net salaries
   For x = 0 To 6
       decTaxAmt = decSalAmounts(x, 0) * 0.18
       decSalAmounts(x, 2) = decSalAmounts(x, 0) - decTaxAmt - 425
   Next

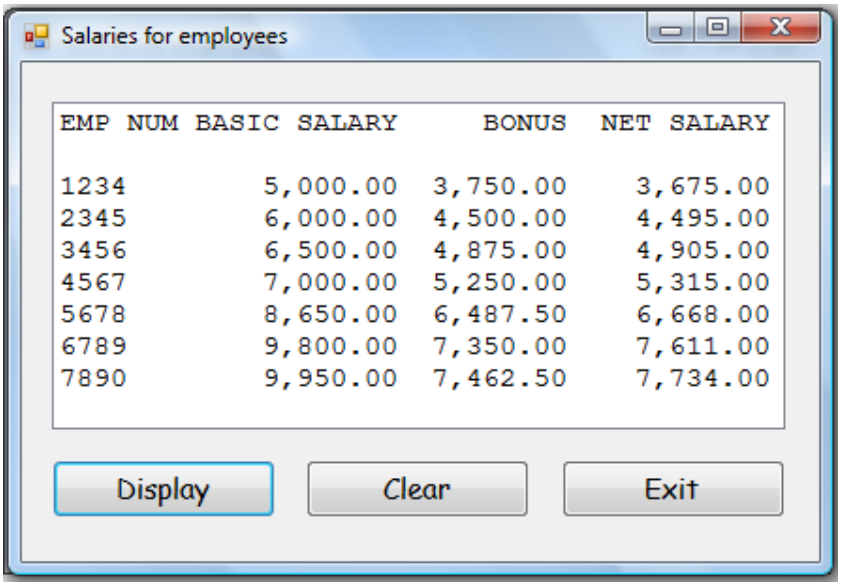
   'Display headings
   lstEmployees.Items.Add("EMP NUM" & "BASIC SALARY".PadLeft(13) & _
                          "BONUS".PadLeft(10) & "NET _
                          SALARY".PadLeft(12))
   lstEmployees.Items.Add(" ")

   'Display array values in a list box
   For x = 0 To 6
       lstEmployees.Items.Add(strEmpNo(x).PadRight(7) & _
                  decSalAmounts(x, 0).ToString("N2").PadLeft(13) & _
                  decSalAmounts(x, 1).ToString("N2").PadLeft(10) & _
                  decSalAmounts(x, 2).ToString("N2").PadLeft(12))
   Next
End Sub
```

```
Private Sub btnClear_Click(. . .) Handles btnClear.Click
   lstEmployees.Items.Clear()
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Output:



CHAPTER 4
QUESTIONS & ANSWERS
Sequential Text Files

QUESTIONS:

QUESTION 1

Indicate whether the following statements are TRUE or FALSE. Provide a reason for your answer.

1.16 An input file should be closed as soon as all data has been read from it.
1.17 It is possible to write data to a specific position between other data on an input file.
1.18 Every file that is used in the program must be opened before it can be used.
1.19 The following code in an algorithm, to read data from a file, is correct:

```
Do until eof
    EmployeeData = read(employees.txt)
    Display EmployeeData
Loop
```

1.20 In VB.NET, the streamReader object enables you to read data from a sequential file.
1.21 The ReadLine Method must include the file name.
1.22 The Peek Method will return a number 0 if the end of the file has been encountered.
1.23 The Write Method and WriteLine Method will always produce exactly the same result.
1.24 It is impossible to modify existing values in a text file.
1.25 The Redim statement may increase the size of an array but it may not decrease the size of an array.
1.26 The Preserve keyword is compulsory in the ReDim statement.
1.27 The following declaration is invalid:

```
Dim intNumbers() As Integer = {}
```

1.28 A sequential access file is often referred to as a text file.
1.29 The string returned by the ReadLine method returns the newline character.
1.30 The data in a sequential access file is always accessed in a consecutive order from the beginning of the file to the end of the file.

QUESTION 2

In each of the following cases, encircle the letter of the most correct answer.

2.16 Which of the following statements will be a valid way of opening the file `student.txt` to read data from it? You may assume the streamReader variable is called `inpStudFile`.

f. inpStudFile = OpenText(student.txt)
g. inpStudFile = File.IO.OpenText(student.txt)
h. inpStudFile = IO.File.OpenText(student.txt)
i. inpStudFile = IO.File.OpenCreate(student.txt)
j. inpStudFile = IO.File.OpenAppend(student.txt)

2.17 Which of the following statements read a line of text from a sequential access file (stream reader variable: inEmpFile) and assigns the line (excluding the newline character) to the variable called strEmpName ?

a. inEmpFile.Read(strEmpName)
b. strEmpName = inEmpFile.ReadLine
c. inEmpFile .ReadLine(strEmpName)
d. strEmpName = inEmpFile.Read(Line)
e. None of the above

2.18 To avoid a runtime error if the computer tries to open a file that does not exist the _____ method can be used to test whether the file exists or not.

a. OpenText
b. OpenExists
c. Exists
d. OpenAppend
e. None of the above

2.19 You can use the Close method to close files opened for _____

a. Append
b. Input
c. Output
d. All of the above
e. None of the above

2.20 If the file you want to open exists, the _____ method erases the file's contents

a. AddText
b. AppendText
c. CreateText
d. InsertText
e. OpenText

2.21 What entity best describes value of multiple elements of the same data type stored under the same name:

a. Variable
b. Structure
c. Array
d. User Defined Data Type
e. None of the above

2.22 Which of the following is not true about a structure?

a. It consists of a number of members
b. All the members must be of the same data type
c. It must end with the End Structure clause
d. It is a user defined data type
e. A structure may be sent as a parameter to an independent sub procedure

2.23 Given the following code, which of the choices below would be used to access the phone number of the 3rd vendor?

```
Structure VendorInfo
        Public Number As String
        Public Name As String
        Public Phone As String
EndStructure
Dim Vendor(5) As VendorInfo
```

a. Vendor(2).Phone
b. VendorInfo(2).Phone
c. Vendor(2)
d. VendorInfo(2)
e. None of the above

2.24 A library keeps track of all of the books in a database file. Author, Title, and Subject are _____ in the database table.

a. Files
b. Records
c. Fields
d. Keys
e. Structures

2.25 Each row in a database table represents a _____

a. File
b. Field
c. Record
d. Key field
e. None of the above

2.26 A preschool has decided to computerize its information about each child. A sequential access file will be created to store the data. All of the information about one child is stored in one _____

a. File
b. Record
c. Field
d. Key field
e. None of the above

2.27 The method to test if the end of a sequential access file has been encountered is

a. eof
b. Peek
c. NoNewLine
d. EndFile
e. None of the above

2.28 The values in the array called intNumbers will be the following after the ReDim statement has been applied to the array.

```
Dim intNumbers() As Integer = {5, 5, 5, 5, 5}
ReDim Preserve intNumbers(5)
```

a. 5, 5, 5, 5, 5
b. 5, 5, 5, 5, 5, 5
c. 0, 0, 0, 0, 0, 0
d. 5, 5, 5, 5, 5, 0
e. 5

QUESTION 3

Provide the VB.NET code for the following questions

3.1 Open an existing text file called "**diary.txt**" for input

3.2 Suppose a structure called Entry is given

```
Structure Entry
        Private strFirstName as String
        Private strLastName as String
        Private strE-mail as String
        Private strTelephone as String
End Structure
```

Write the code to read 10 lines of text from the file into an array of this structure data type. Every field on a text line is separated by a comma and the telephone number is ended by a full stop e.g.
Freddy,Gomes,GomesF@gmail.com,0824154433.

You may assume that the file exists and that there will be exactly 10 text lines on the file.

QUESTION 4

Create a structure called Student, containing the following members:
Student number,
student name,
course code,
number of subjects passed,
level.

Use your structure to declare a structure variable called FirstYear.

QUESTION 5

Write the necessary statements to test if the file "c:\numbers.txt" exists. If it does, display the number of lines contained in the file on the label called lblNumber, otherwise a suitable error message must be displayed in a message box.

QUESTION 6

Write VB.NET programs to solve the following problems.

6.1 The following problem uses the same file and structure as example 4.14 in chapter 4:

All the employees in the company are part time employees and pay 25% tax. Calculate the net monthly salary for every employee and display the employee number and net monthly salary for every employee in a list box. Display suitable headings and display the output in two columns in the list box. The file is not changed and therefore a new file need not be created after all the employees have been read and processed.

6.2 The following problem uses the same file and structure as the example in 4.14 in chapter 4:

Append new employees to the end of the file. For every new employee enter the employee number, name, gender, department and annual salary via input boxes. Validate that the department is only one character in length, that the gender is only F or M and that the annual salary is a decimal value greater than 0. If not, display a suitable error message and keep on entering the value until it is correct. Display the number, name and department code of every new employee in columns on a list box. Provide for suitable headings.

6.3 A file "c:\products.txt" contains the following for all the products in a store:

Product code
Product description
Number in stock
Unit price

An asterisk indicates the end of each field.
E.g. STA001*Flip File*320*12.99*

Write a VB.NET program to increase all the prices in the store by 7.5% and to display the product code, old price and new price of all products in the store on a list box.

ANSWERS:

QUESTION 1

1.1 TRUE
Yes, a file should be closed as soon as possible to prevent losing information.

1.2 FALSE
You cannot write data between other data on an input file. However, you can read all the text lines from the input file into an array, then insert a new element between other elements in the array and then create this file again as an output file.

1.3 TRUE
Yes, this is done to connect the external text file to the internal program file stream object in order to use it in the program. It will also identify if the file is opened for input, output or append.

1.4 FALSE
The first record must be read before the loop, the next record must be read just before the next execution of the loop. Therefore, the pseudo code must change as follows in order for it to be correct.

```
EmployeeData = read(employees.txt)
Do until eof
    display EmployeeData
    EmployeeData = read(employees.txt)
Loop
```

1.5 TRUE
Yes, when it is created and stored in a variable in the internal memory of the computer, it can be used to read data from a sequential access file.

1.6 FALSE
The ReadLine Method does not include the file name as it uses the file associated with the StreamReader variable.

1.7 FALSE
The Peek Method will return a number -1 if the end of the file has been encountered.

1.8 FALSE
The Write Method will not write a newline character at the end of the data whereas the WriteLine Method will.

1.9 FALSE
Values in the file can be modified if the file is first opened as input and all the text lines are read into an array. The values can now be modified in the array and then written back to the file by creating a new text file with the same name.

1.10 FALSE
The ReDim statement can increase or decrease the size of an array.

1.11 FALSE
It may be omitted, in which case all existing values in the array will be lost.

1.12 FALSE
In this case the array is declared, but it contains no elements. Elements can be added to the array as long as the ReDim statement is used in each case to increase the size of the array first.

1.13 TRUE
Because a sequential access file consists of lines of text data.

1.14 FALSE
The ReadLine Method reads until the newline character is detected, but it will not include the newline character in the string that is returned.

1.15 TRUE
Data is stored in a sequential manner and can also only be accessed in a sequential manner. In order to read the 4th line of text, one would have to read through the previous 3 lines of text as well.

QUESTION 2

2.1 c
2.2 b
2.3 c
2.4 d
2.5 c
2.6 c
2.7 b
2.8 a
2.9 c
2.10 c
2.11 e
2.12 b
2.13 d

QUESTION 3

Provide the VB.NET code for the following questions

Open an existing text file called "**diary.txt**" for input.

```
Dim inputDiary As IO.StreamReader
inputDiary = IO.File.OpenText("diary.txt")
```

Suppose a structure called Entry is given

```
Structure Entry
    Private strFirstName as String
    Private strLastName as String
    Private strE-mail as String
    Private strTelephone as String
End Structure
```

Write the code to read 10 lines of text from the file into an array of this structure data type. Every field on a text line is separated by a comma and the telephone number is ended by a full stop e.g.
Freddy,Gomes,GomesF@gmail.com,0824154433.

You may assume that the file exists and that there will be exactly 10 lines of text on the file.

```
Dim strDiary(9) As Entry
Dim intIndex As integer = 0
Dim strOneLine As String
Dim intPosition As Integer
```

```
For intIndex = 0 to 10

      strOneLine = inputDiary.ReadLine

      intPosition = strOneLine.IndexOf(",")
      strDiary(intIndex).strFirstName = _
                        strOneLine.Substring(0, intPosition)
      strOneLine = strOneLine.Remove(0, intPosition + 1)

      intPosition = strOneLine.IndexOf(",")
      strDiary(intIndex).strLastName = _
                        strOneLine.Substring(0, intPosition)
      strOneLine = strOneLine.Remove(0, intPosition + 1)

      intPosition = strOneLine.IndexOf(",")
      strDiary(intIndex).strE-mail = _
                        strOneLine.Substring(0, intPosition)
      strOneLine = strOneLine.Remove(0, intPosition + 1)

      intPosition = strOneLine.IndexOf(".")
      strDiary(intIndex).strTelephone = _
                        strOneLine.substring(0, intPosition)

      intIndex = intIndex + 1

Next
```

QUESTION 4

Create a structure called Student, containing the following members:
- Student number,
- student name,
- course code,
- number of subjects passed,
- level.

Use your structure to declare a structure variable called FirstYear.

```
Structure Student
      Public strStNumber As String
      Public strStName As String
      Public strCourseCode As String
      Public intSubjPassed As Integer
      Public intLevel As Integer
End Structure

Dim FirstYear As Student
```

QUESTION 5

Write the necessary statements to test if the file "c:\numbers.txt" exists. If it does, display the number of lines contained in the file on the label called lblNumber, otherwise a suitable error message must be displayed in a message box.

```
Dim intCount As Integer = 0
Dim inputNumbers As IO.StreamReader
inputNumbers = IO.File.OpenText("c:\numbers.txt")
If IO.File.Exists("c:\numbers.txt") Then
      inputNumbers.ReadLine
      Do Until inputNumbers.Peek <> -1
            intCount = intCount + 1
            inputNumbers.ReadLine
      Loop
      lblNumber.Text = "There are "  & intCount & _
                       " lines of text in the file"
Else
      MessageBox.Show("The file does not exist")
End If
```

QUESTION 6

Solve the following problems by first doing the planning and then writing the VB.NET program.

6.1 The following problem uses the same file and structure as ~~in~~ the example in 4.14:

All the employees in the company are part time employees and pay 25% tax. Calculate the net monthly salary for every employee and display the employee number and net monthly salary for every employee in a list box. Display suitable headings and display the output in two columns in the list box. The file is not changed and therefore a new file need not be created after all the employees have been read and processed.

Code in the code editor:

```
Public Class Employees

    Structure Employee
        Public strENumber As String
        Public strEName As String
        Public chaGender As Char
        Public chaDept As Char
        Public decAnnSal As Decimal
    End Structure
    Private strEmployee() As Employee = {}
    Private intArrayIndex As Integer = 0
    Private blnValidProcessing As Boolean = True
```

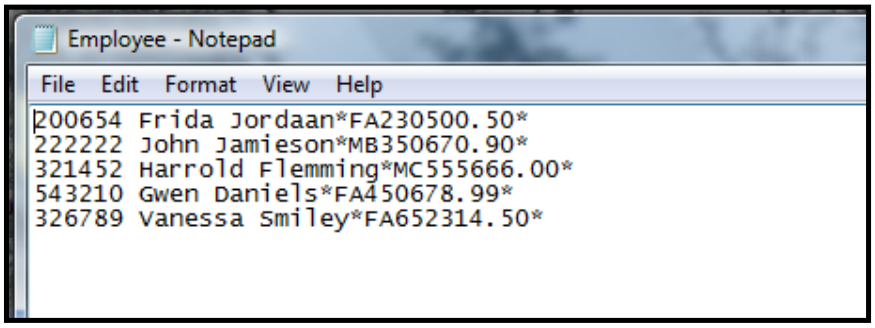
```
Private Sub Employees_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
   Dim InEmpFile As System.IO.StreamReader
   Dim strOneCompleteLine As String
   Dim intPosition As Integer
   If System.IO.File.Exists("C:\Employee.txt") Then
      InEmpFile = System.IO.File.OpenText("C:\Employee.txt")
      strOneCompleteLine = InEmpFile.ReadLine
      Do While InEmpFile.Peek <> -1
         ReDim Preserve strEmployee(intArrayIndex)
         intPosition = strOneCompleteLine.IndexOf(" ")
         strEmployee(intArrayIndex).strENumber = _
                     strOneCompleteLine.Substring(0, intPosition)
         strOneCompleteLine = strOneCompleteLine.Remove(0, intPosition)
         intPosition = strOneCompleteLine.IndexOf("*")
         strEmployee(intArrayIndex).strEName = _
                     strOneCompleteLine.Substring(0, intPosition)
         strOneCompleteLine = strOneCompleteLine.Remove(0, intPosition + 1)
         strEmployee(intArrayIndex).chaGender = _
                     strOneCompleteLine.Substring(0, 1)
         strOneCompleteLine = strOneCompleteLine.Remove(0, 1)
         strEmployee(intArrayIndex).chaDept = _
                     strOneCompleteLine.Substring(0, 1)
         strOneCompleteLine = strOneCompleteLine.Remove(0, 1)
         intPosition = strOneCompleteLine.IndexOf("*")
         strEmployee(intArrayIndex).decAnnSal = _
                     strOneCompleteLine.Substring(0, intPosition)
         strOneCompleteLine = InEmpFile.ReadLine
         intArrayIndex += 1
      Loop
      intArrayIndex = intArrayIndex - 1
      InEmpFile.Close()
      MessageBox.Show("Array elements loaded")
   Else
      MessageBox.Show("File does not exit, you must create it first")
      blnValidProcessing = False
   End If
End Sub

Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnExit.Click
   Close()
End Sub

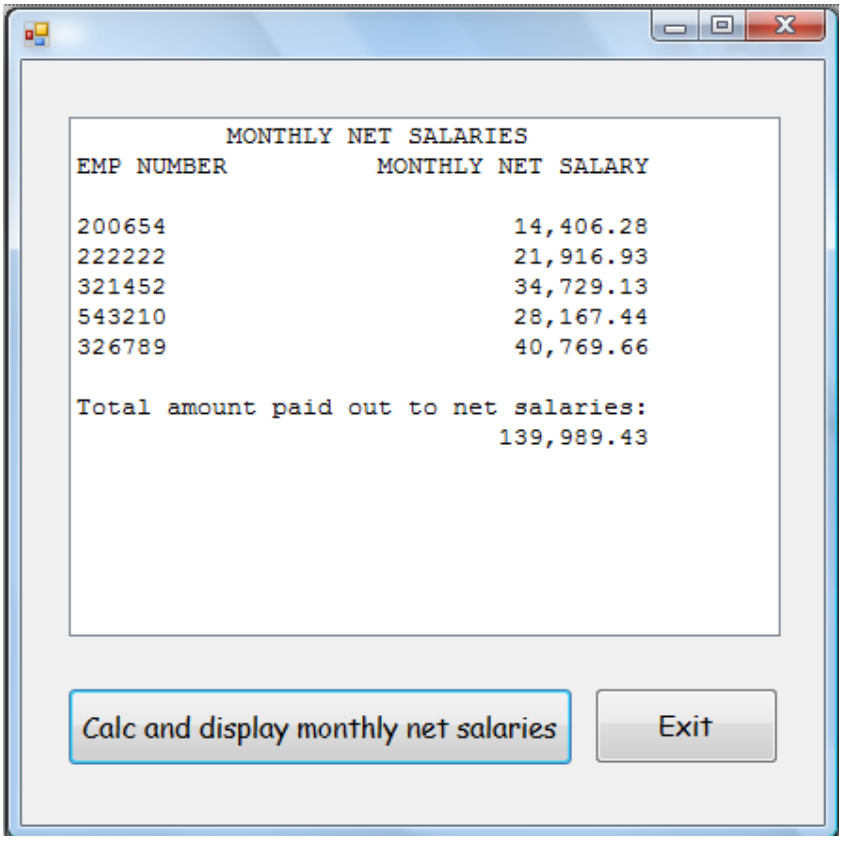
Private Sub btnCalc_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCalc.Click
   Dim decMonthlyGrossSal As Decimal
   Dim decTaxAmount As Decimal
   Dim decMonthlyNetSal As Decimal
   Dim decTotalPaid As Decimal = 0
   If blnValidProcessing Then
      lstDisplay.Items.Add("          MONTHLY NET SALARIES")
      lstDisplay.Items.Add("EMP NUMBER".PadRight(20) & _
                           "MONTHLY NET SALARY".PadLeft(18))
      lstDisplay.Items.Add(" ")
```

```
        For x As Integer = 0 To intArrayIndex
            decMonthlyGrossSal = strEmployee(x).decAnnSal / 12
            decTaxAmount = decMonthlyGrossSal * 0.25
            decMonthlyNetSal = decMonthlyGrossSal - decTaxAmount
            decTotalPaid += decMonthlyNetSal
            lstDisplay.Items.Add(strEmployee(x).strENumber.PadRight(20) _
                    & decMonthlyNetSal.ToString("N2").PadLeft(18))
        Next
        lstDisplay.Items.Add(" ")
        lstDisplay.Items.Add("Total amount paid out to net salaries:")
        lstDisplay.Items.Add(decTotalPaid.ToString("N2").PadLeft(38))
    Else
        MessageBox.Show("Sorry, you must create the file first")
    End If
End Sub
End Class
```

Example of the file "C:\Employee.txt" before the program is executed:



Output:



6.2 The following problem uses the same file and structure as the example in 4.14 in chapter 4:
Append new employees to the end of the file. For every new employee enter the employee number, name, gender, department and annual salary via input boxes. Validate that the department is only one character in length, that the gender is only F or M and that the annual salary is a decimal value greater than 0. If not, display a suitable error message and keep on entering the value until it is correct. Display the number, name and department code of every new employee in columns on a list box. Provide for suitable headings.

Code in the code editor:

```
Public Class Employees

    Structure Employee
        Public strENumber As String
        Public strEName As String
        Public chaGender As Char
        Public chaDept As Char
        Public decAnnSal As Decimal
    End Structure
    Private strEmployee() As Employee = {}
    Private intArrayIndex As Integer = 0
    Private blnValidProcessing As Boolean = True

Private Sub Employees_Load(. . .) Handles MyBase.Load
   Dim InEmpFile As System.IO.StreamReader
   Dim strOneCompleteLine As String
   Dim intPosition As Integer
   If System.IO.File.Exists("C:\Employee.txt") Then
      InEmpFile = System.IO.File.OpenText("C:\Employee.txt")
      strOneCompleteLine = InEmpFile.ReadLine
      Do While InEmpFile.Peek <> -1
         ReDim Preserve strEmployee(intArrayIndex)
         intPosition = strOneCompleteLine.IndexOf(" ")
         strEmployee(intArrayIndex).strENumber = _
                     strOneCompleteLine.Substring(0, intPosition)
         strOneCompleteLine = strOneCompleteLine.Remove(0, intPosition + 1)
         intPosition = strOneCompleteLine.IndexOf("*")
         strEmployee(intArrayIndex).strEName = _
         strOneCompleteLine.Substring(0, intPosition)
         strOneCompleteLine = strOneCompleteLine.Remove(0, intPosition + 1)
         strEmployee(intArrayIndex).chaGender = _
                     strOneCompleteLine.Substring(0, 1)
         strOneCompleteLine = strOneCompleteLine.Remove(0, 1)
         strEmployee(intArrayIndex).chaDept = _
                     strOneCompleteLine.Substring(0, 1)
         strOneCompleteLine = strOneCompleteLine.Remove(0, 1)
         intPosition = strOneCompleteLine.IndexOf("*")
         strEmployee(intArrayIndex).decAnnSal = _
                     strOneCompleteLine.Substring(0, intPosition)
         strOneCompleteLine = InEmpFile.ReadLine
         intArrayIndex += 1
      Loop
      intArrayIndex = intArrayIndex - 1
      InEmpFile.Close()
      MessageBox.Show("Array elements loaded")
   Else
       MessageBox.Show("File does not exit, you must create it first")
        blnValidProcessing = False
```

```
   End If
End Sub
Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnExit.Click
   Close()
End Sub

Private Sub btnNew_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNew.Click
   Dim strENum As String
   Dim strName As String
   Dim strGender As String
   Dim strSal As String
   Dim strDepartment As String
   Dim chaGendr As Char
   Dim chaDepartment As Char
   Dim decSal As Decimal
   Dim blnValid As Boolean = False
   Dim intNoNewemployees As Integer = 0
   If blnValidProcessing Then
      lstDisplay.Items.Add("            NEW EMPLOYEES")
      lstDisplay.Items.Add("EMP NUMBER".PadRight(12) & _
                           "EMP NAME".PadRight(20) & "DEPT")
      lstDisplay.Items.Add(" ")
      strENum = InputBox("Please enter the number of the employee", " ")
      Do While strENum <> ""
         strName = InputBox("Please enter name of employee", " ")
         Do
           strGender = InputBox("Please enter gender - M or F", " ")
           blnValid = Char.TryParse(strGender.ToUpper, chaGendr)
           If chaGendr.ToString.ToUpper = "F" Or _
              chaGendr.ToString.ToUpper = "M" Then
              blnValid = True
           Else
              blnValid = False
              MessageBox.Show("It can only be M or F - please re-enter")
           End If
         Loop Until blnValid
         Do
           strDepartment = InputBox(_
                "Please enter the department code as one character", " ")
           blnValid = Char.TryParse(strDepartment.ToUpper, chaDepartment)
           If Not blnValid Then
              MessageBox.Show("It must be one character - please re-enter")
           End If
         Loop Until blnValid
         Do
           strSal = InputBox("Please enter the annual salary", " ")
           blnValid = Decimal.TryParse(strSal, decSal)
           If Not blnValid Then
              MessageBox.Show("It must be a decimal value - re-enter")
           Else
              If decSal <= 0 Then
                 MessageBox.Show("It can't be 0 or negative - re-enter")
              End If
           End If
         Loop Until blnValid And decSal > 0
         lstDisplay.Items.Add(strENum.PadRight(12) & strName.PadRight(20) _
                              & chaDepartment.ToString.PadLeft(4))
         intArrayIndex = intArrayIndex + 1
         ReDim Preserve strEmployee(intArrayIndex)
```

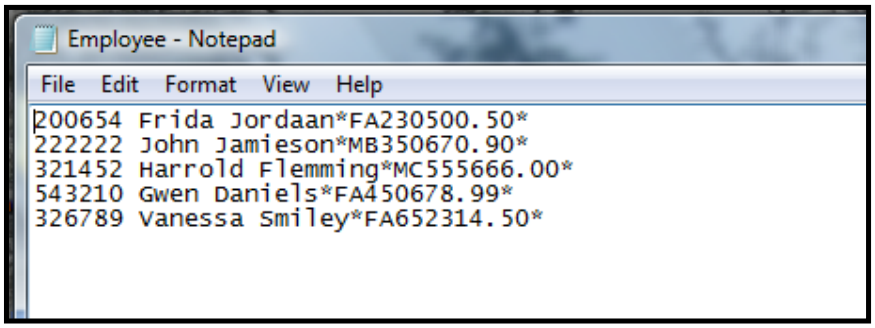
```
            strEmployee(intArrayIndex).strENumber = strENum
            strEmployee(intArrayIndex).strEName = strName
            strEmployee(intArrayIndex).chaGender = chaGendr
            strEmployee(intArrayIndex).chaDept = chaDepartment
            strEmployee(intArrayIndex).decAnnSal = decSal
            MessageBox.Show("Thank You, employee added")
            intNoNewemployees += 1
            strENum = InputBox("Please enter number of the next employee", _
                               "Click on calncel to end")
        Loop
        If intNoNewemployees = 0 Then
           lstDisplay.Items.Add("No new employees entered")
        End If
    Else
       MessageBox.Show("Sorry, you must create the file first")
    End If
End Sub

Private Sub Employees_FormClosed(ByVal sender As Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
    If blnValidProcessing Then
       Dim OutEmpFile As System.IO.StreamWriter
       Dim strOutLine As String
       OutEmpFile = System.IO.File.CreateText("C:\Employee.txt")
       For x As Integer = 0 To intArrayIndex
           strOutLine = strEmployee(x).strENumber & " " & _
                        strEmployee(x).strEName & "*" & _
                        strEmployee(x).chaGender & _
                        strEmployee(x).chaDept & _
                        strEmployee(x).decAnnSal.ToString("F2") & "*"
           OutEmpFile.WriteLine(strOutLine)
       Next
       OutEmpFile.Close()
       MessageBox.Show("Thank you, new employees added")
    End If
End Sub

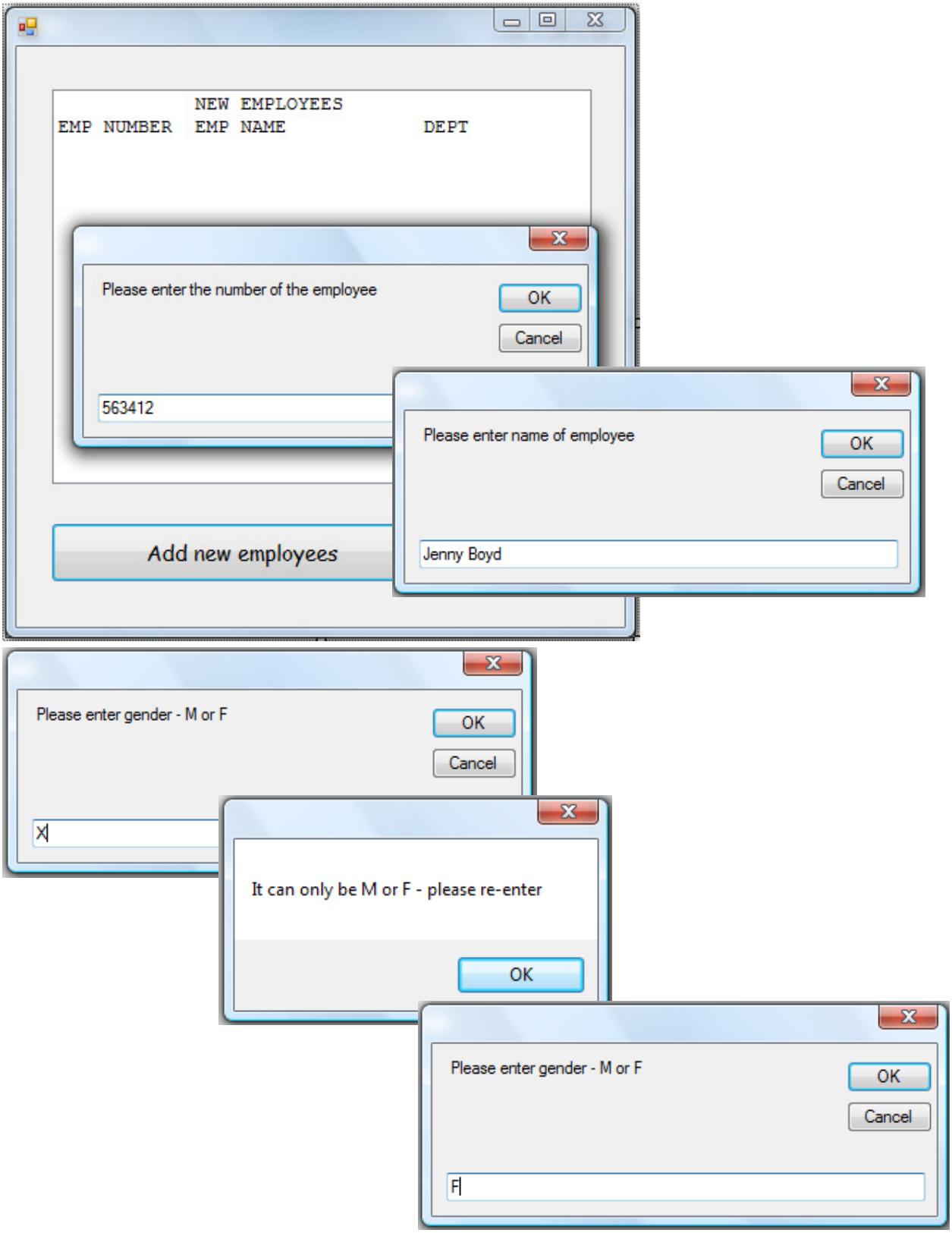
End Class
```

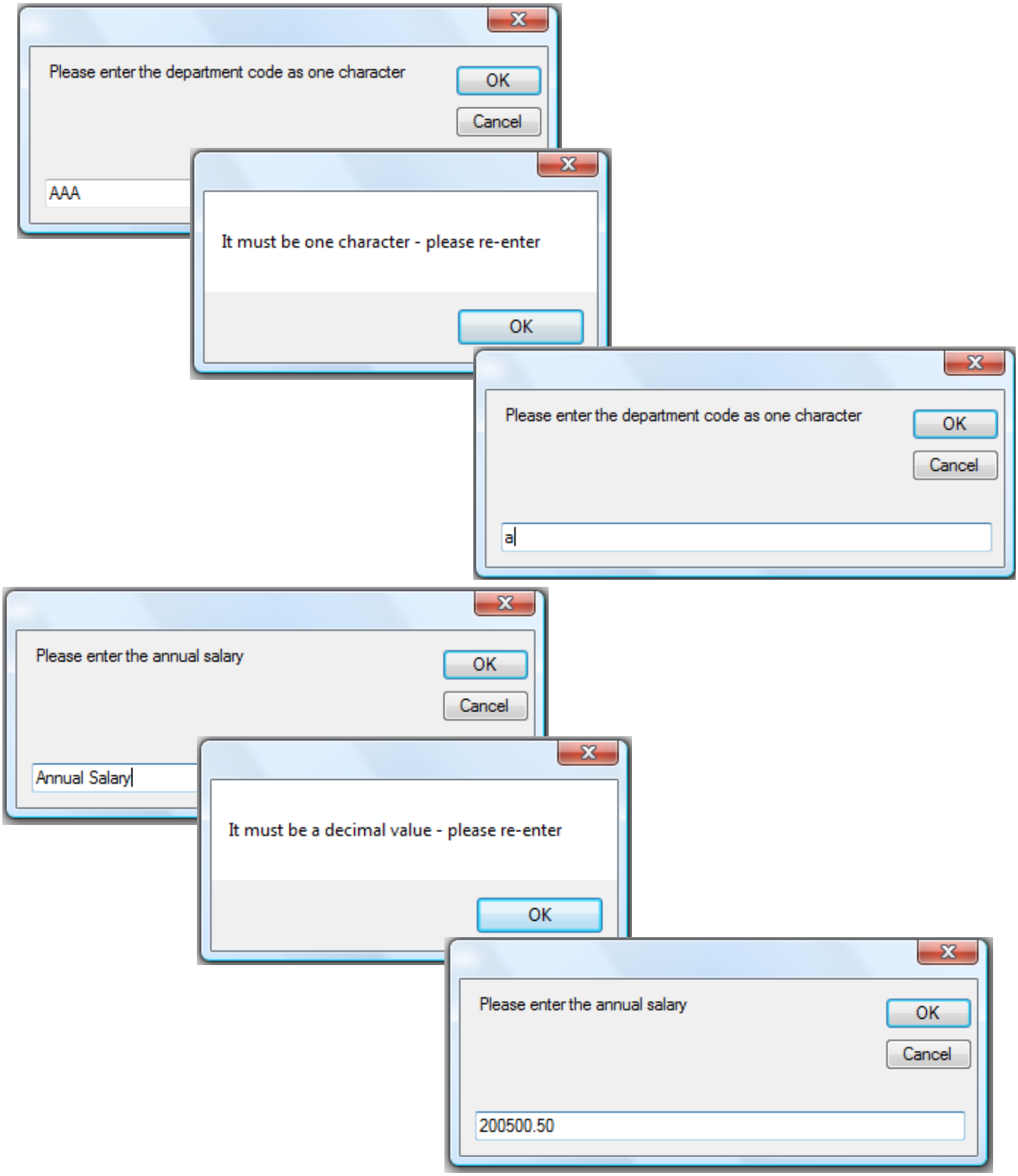
Output:

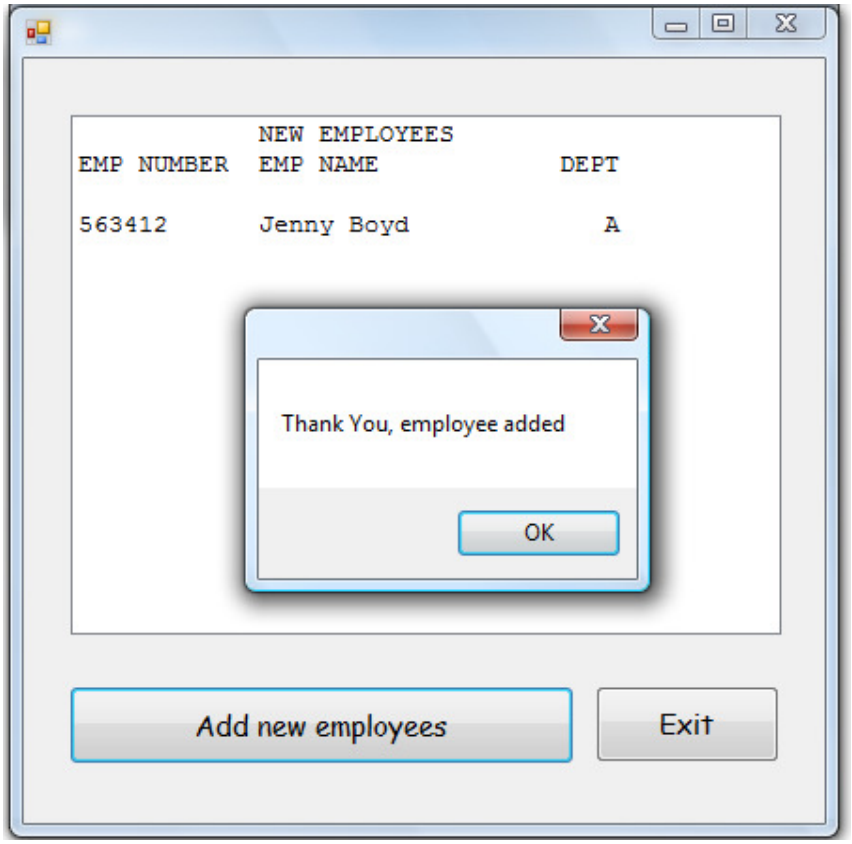
Example of the file "C:\Employee.txt" before the program is executed:



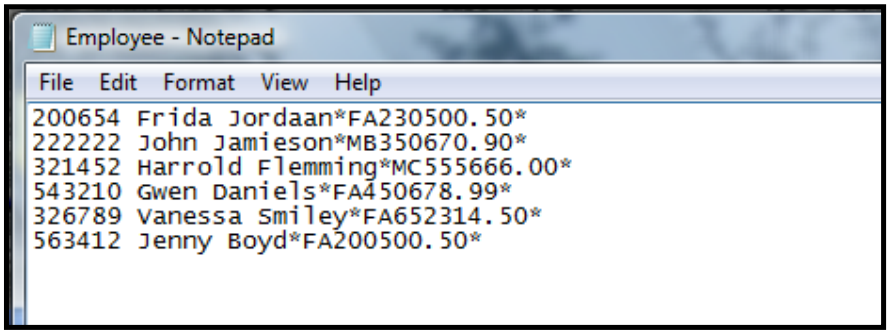
Examples of new input values entered and displayed on the list box:




Please enter the department code as one character
OK
Cancel
AAA
It must be one character - please re-enter
OK
Please enter the department code as one character
OK
Cancel
a
Please enter the annual salary
OK
Cancel
Annual Salary
It must be a decimal value - please re-enter
OK
Please enter the annual salary
OK
Cancel
200500.50



Contents of the file "C:\Employee.txt" after the program has executed:



6.3 A file "c:\products.txt" contains the following for all the products in a store:

Product code
Product description
Number in stock
Unit price

An asterisk indicates the end of each field.
E.g. STA001*Flip File*320*12.99*

Write a VB.NET program to increase all the prices in the store by 7.5% and to display the product code, old price and new price of all products in the store on a list box.

Code in the code editor:

```
Public Class ProductsEx
   Structure Products
      Public strProdCode As String
      Public strProdDesc As String
      Public intNumStock As Integer
      Public decPrice As Decimal
   End Structure
   Private strProducts() As Products = {}
   Private intArrayIndex As Integer = 0
   Private blnValidProcessing As Boolean = True

Private Sub ProductsEx_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
   Dim InProdFile As System.IO.StreamReader
   Dim strOneCompleteLine As String
   Dim intPosition As Integer
   If System.IO.File.Exists("C:\Prod.txt") Then
      InProdFile = System.IO.File.OpenText("C:\Prod.txt")
      strOneCompleteLine = InProdFile.ReadLine
      Do While InProdFile.Peek <> -1
         ReDim Preserve strProducts(intArrayIndex)
         intPosition = strOneCompleteLine.IndexOf("*")
         strProducts(intArrayIndex).strProdCode = _
                     strOneCompleteLine.Substring(0, intPosition)
         strOneCompleteLine = strOneCompleteLine.Remove(0, intPosition + 1)
         intPosition = strOneCompleteLine.IndexOf("*")
         strProducts(intArrayIndex).strProdDesc = _
                     strOneCompleteLine.Substring(0, intPosition)
         strOneCompleteLine = strOneCompleteLine.Remove(0, intPosition + 1)
         intPosition = strOneCompleteLine.IndexOf("*")
         strProducts(intArrayIndex).intNumStock = _
                     strOneCompleteLine.Substring(0, intPosition)
         strOneCompleteLine = strOneCompleteLine.Remove(0, intPosition + 1)
         intPosition = strOneCompleteLine.IndexOf("*")
         strProducts(intArrayIndex).decPrice = _
                     strOneCompleteLine.Substring(0, intPosition)
         strOneCompleteLine = InProdFile.ReadLine
         intArrayIndex += 1
      Loop
      intArrayIndex = intArrayIndex - 1
      InProdFile.Close()
      MessageBox.Show("Array elements loaded")
   Else
      MessageBox.Show("File does not exit, you must create it first")
      blnValidProcessing = False
   End If
End Sub

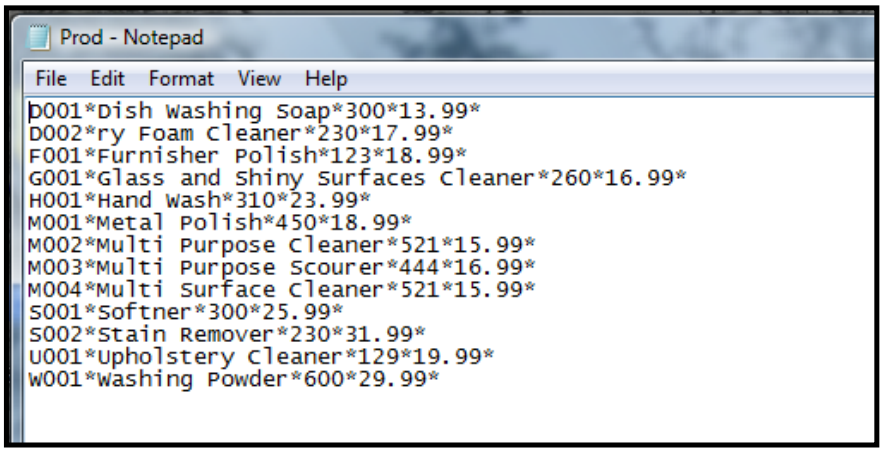
Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnExit.Click
   Close()
End Sub
```

```
Private Sub btnIncreaseShow_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnIncreaseShow.Click
   Dim decOldPrice, decNewPrice As Decimal
   If blnValidProcessing Then
      lstDisplay.Items.Add("PROD CODE".PadRight(10) & _
                 "OLD PRICE".PadLeft(15) & "NEW PRICE".PadLeft(15))
      lstDisplay.Items.Add(" ")
      For x As Integer = 0 To intArrayIndex
          decOldPrice = strProducts(x).decPrice
          decNewPrice = decOldPrice * 1.075
          strProducts(x).decPrice = decNewPrice
          lstDisplay.Items.Add(strProducts(x).strProdCode.PadRight(10) _
                           & decOldPrice.ToString("N2").PadLeft(15) & _
                           decNewPrice.ToString("N2").PadLeft(15))
      Next
   Else
      MessageBox.Show("No Product file")
   End If
End Sub

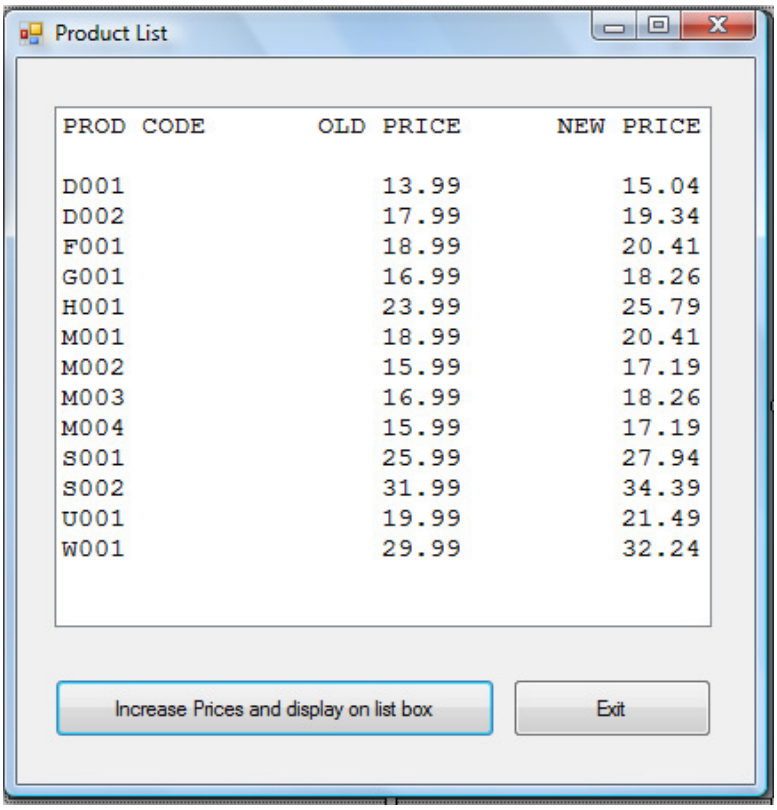
Private Sub ProductsEx_FormClosed(ByVal sender As Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
   If blnValidProcessing Then
      Dim OutProdFile As System.IO.StreamWriter
      Dim strOutLine As String
      OutProdFile = System.IO.File.CreateText("C:\Prod.txt")
      For x As Integer = 0 To intArrayIndex
          strOutLine = strProducts(x).strProdCode & "*" & _
                       strProducts(x).strProdDesc & "*" & _
                       strProducts(x).intNumStock & "*" & _
                       strProducts(x).decPrice.ToString("F2") & "*"
          OutProdFile.WriteLine(strOutLine)
      Next
      OutProdFile.Close()
      MessageBox.Show("Thank you, Price increases updated")
   End If
End Sub

End Class
```

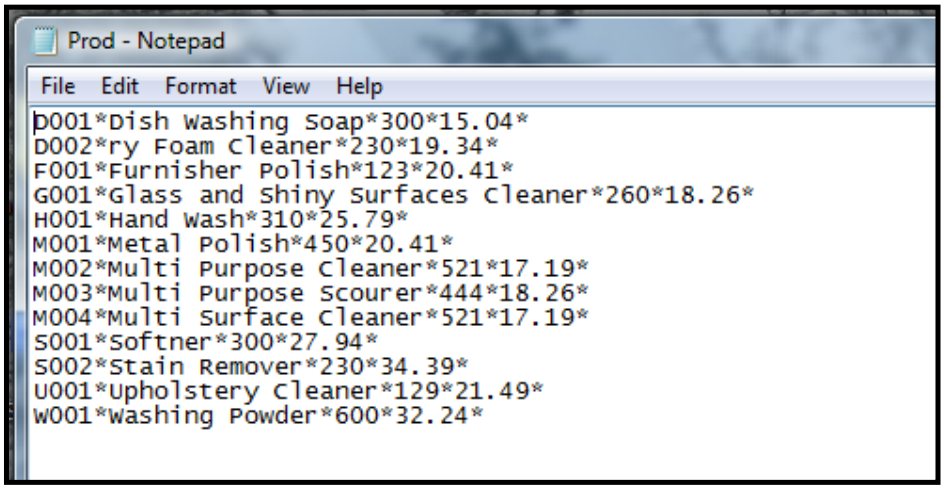
Example of the file "C:\Prod.txt" before the program is executed:



Output:



Example of the file "C:\Prod.txt" before the program is executed:



CHAPTER 5
QUESTIONS & ANSWERS
Advanced Features and Built-In Functions

QUESTIONS:

QUESTION 1

Indicate whether the following statements are TRUE or FALSE. Provide a reason for your answer.

1.1 When using the Round method, the result will always be an integer.
1.2 The square root method will return the same value as the argument multiplied by 0.5.
1.3 It is possible to return the absolute value of a positive or a negative number.
1.4 More than one mathematical method may be used in one equation.
1.5 The method to raise a number to a specific power is called *Exp*.
1.6 It is not valid to calculate the square of a number by multiplying it by itself.
1.7 The square root of a number will be of the type double.
1.8 When obtaining a random number from a specific range, this range must always start at 0 (zero)
1.9 To obtain a random number, at least 5 lines of coding are needed.
1.10 The Pmt function may be used to determine how much money must be saved per month for a specific number of months to enable the user to save a known amount of money.
1.11 The interest rate per period is a compulsory entry when calculating a payment
1.12 It is possible to obtain the square root of any integer.
1.13 The DateTimePicker may be used to calculate yesterday's date.
1.14 The DateTimePicker can display the hours and minutes, but not the seconds.
1.15 The DateTimePicker can display the date in the format m/d/yyyy where m and d can each represent one or two digits.

QUESTION 2

In each of the following cases, encircle the letter of the most correct answer.

2.1 When more than one method is used an equation, the calculations are done
  a. from left to right
  b. from the inside to the outside, taking the brackets in consideration
  c. from the outside to the inside, taking the brackets in consideration
  d. none of the above

2.2 The function that may be used to determine the annual payment of a loan in 10 years time is the _____ method
  a. Pmt
  b. Payment
  c. Instalment
  d. Amt

2.3 When using the absolute method the result may be of the type:
   a. integer
   b. double
   c. string
   d. answers a and b

2.4 When random numbers are selected, _____
   a. all the numbers in the range will be selected when twice the number of numbers in the range are selected
   b. the beginning number and the ending number are never selected
   c. the number of values to be selected may never be more than the number of values in the range
   d. none of the above

2.5 The payment function must always contain a positive
   a. beginning value
   b. final value
   c. period of time
   d. all of the above

2.6 The method used to obtain the square root of a number, is the _____ method
   a. Sqrt( )
   b. Sqroot( )
   c. SquareRoot( )
   d. None of the above

2.7 The DateTimePicker may be used to obtain _____
   a. The system date
   b. This month
   c. This year
   d. All of the above

2.8 The random function can be set to _____
   a. obtain the middle of the range
   b. obtain an unknown number within the range
   c. obtain the highest number in the range
   d. all of the above

2.9 The payment function may be used to _____
   a. obtain the price of an item
   b. calculate the interest payable per day
   c. determine the loan amount
   d. none of the above

2.10 The DateTimePicker can _____
   a. display a calendar used to choose a date
   b. suggest a date
   c. supply the hour, but not the minutes
   d. none of the above

QUESTION 3

Determine the value of Y in each of the following equations:

a. Y = Round (65.456,1)

b. Y = Abs (16+22-54)

c. Y = Round (Pow (A, (B-3)), Sqrt(A)) where A = 9 and B = 5

QUESTION 4

Do the planning and a program to solve the following problem:

Use the range of the 8 integers from 0 to 7. The program must now use the random function to choose a number that is entered by the user, of random integers within this range. The program will count and display how many times each of the integers within the range has been selected. Hint: use an array to count the occurrences.

QUESTION 5

Do the planning and write a program to solve the following problem:

Let the user enter 2 odd integers to serve as the range for obtaining random numbers. Calculate the number in the middle of this range. Now, let the program obtain 7 random numbers and then display which one of these numbers is the nearest to the middle number.

QUESTION 6

Do the planning and write a program to solve the following problem:

George wants to buy a nice big quad, but he needs a deposit of 60% of the price of the quad before the shop will sell it to him. He has to enter the price into your program and then you will have to determine how much his deposit will be and also how much the monthly instalments will be if he pays the remainder over a period of 2 years. He will have to pay 14.8% interest per annum on the outstanding amount. Display the instalment amount on the screen.

QUESTION 7

Write a VB.NET program and use the DateTimePicker to obtain and display today's date from the system. Use today's date to calculate and display the date of a month ago in short format. You may assume today is not in January.

ANSWERS:

QUESTION 1

1.1 FALSE
The number of decimal positions may be specified.
1.2 FALSE
Not logically correct!
1.3 TRUE
1.4 TRUE
1.5 FALSE
It uses the function *Pow*
1.6 FALSE
It is the way to calculate the square
1.7 TRUE
1.8 FALSE
The beginning position can be specified
1.9 FALSE
By using the random function, it is done in one line.
1.10 TRUE
1.11 TRUE
1.12 FALSE
It is not possible to calculate the square root of a negative integer.
1.13 TRUE
It will supply today's date from the system and then yesterday's date can be calculated.
1.14 FALSE
If you set the format to time, it will display hours, minutes and seconds
1.15 TRUE

QUESTION 2

2.1 b
2.2 a
2.3 d
2.4 d
2.5 c
2.6 a
2.7 d
2.8 b
2.9 d
2.10 a

QUESTION 3

Determine the value of Y in each of the following equations:

a. Y = Round (65.456,1)
= 65.5

b. Y = Abs (16+22-54)
= Abs (-16)
= 16

c. Y = Round (Pow (A, (B-3)), Sqrt(A)) where A = 9 and B = 5
Substitute values:
= Round (Pow (9, (5-3)), Sqrt(9))
= Round (Pow(9,2)), 3)
= Round (81,3)
= 81.000

QUESTION 4

Do the planning and a program to solve the following problem:

Use the range of the 8 integers from 0 to 7. The program must now use the random function to choose a number that is entered by the user, of random integers within this range. The program will count and display how many times each of the integers within the range have been selected. Hint: use an array to count the occurrences.

	Description	Type	Name of variable
Input:	number of random numbers	integer	number
	Random number	integer	noRandom
Output:	numbers in the array countArray of the type integer		

Algorithm:

```
CountRandomIntegers

        for i = 0 to 7          ~       initialize the countArray
                countArray(i) = 0
        endfor
        display "How many times do you want to choose random integers?"
        enter number
        for k = 1 to number
                noRandom = random(0, 8)
                countArray(noRandom) = countArray(noRandom) + 1
        endfor


        ~       display the values
                for m = 0 to 7
                                        ~ display on a new line
                display "The integer ", m, " appeared ", countArray(m), " times"
        endfor
end
```

Code in the code editor:

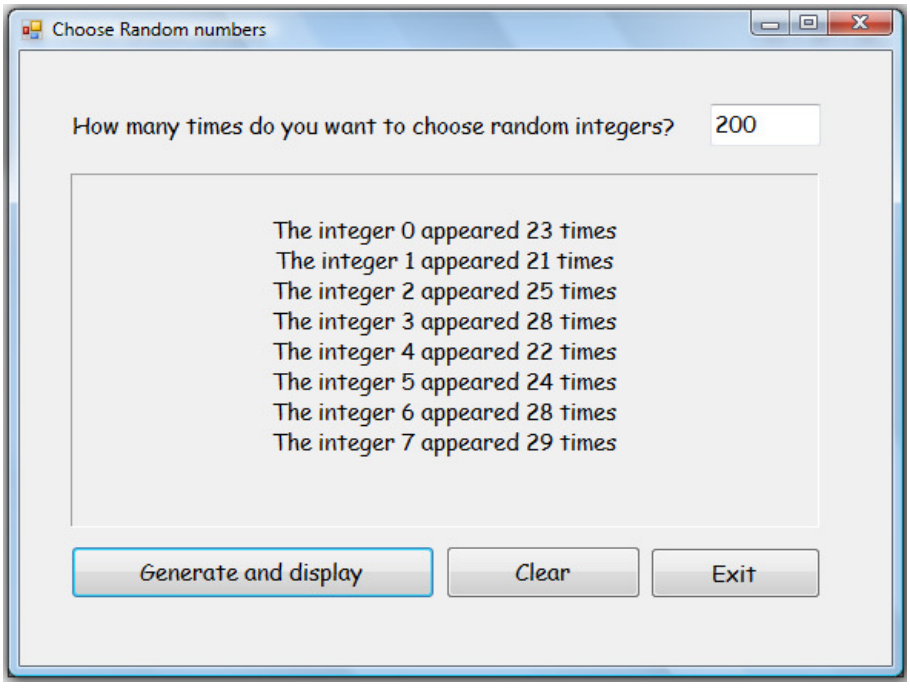
```
Public Class CountRandomInt

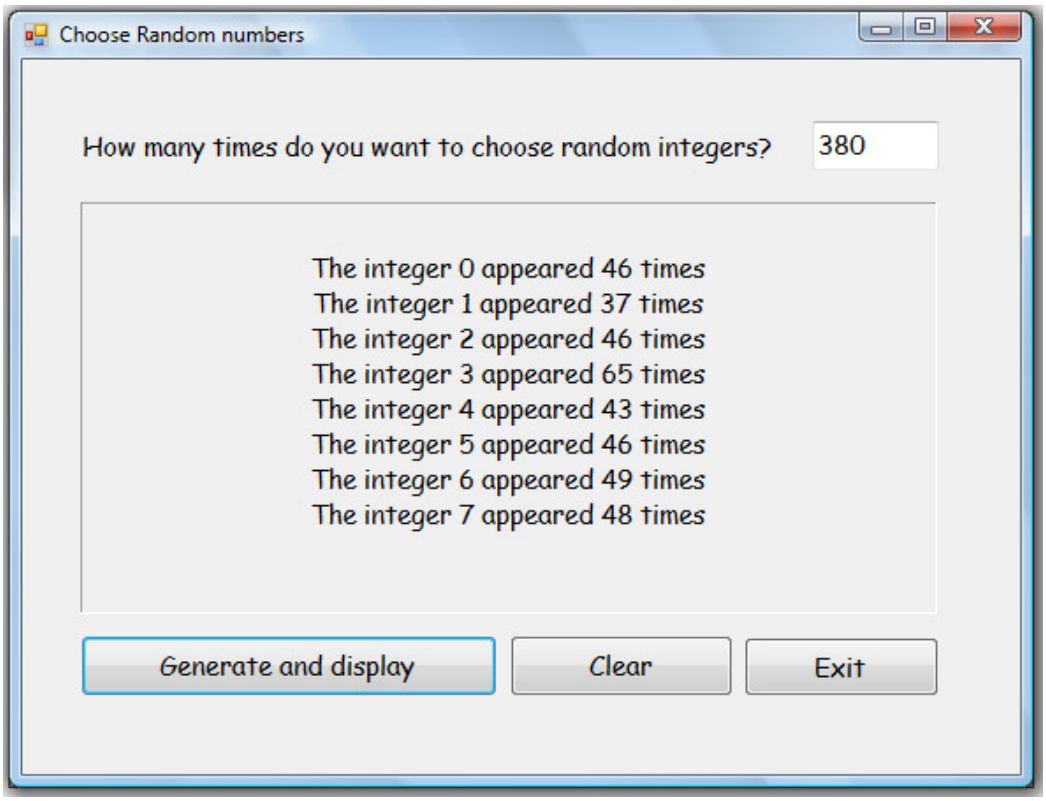
Private Sub btnGenerate_Click(. . .) Handles btnGenerate.Click
   Dim intCountArray(8) As Integer
   Dim intNumber, intRandomNumber As Integer
   Dim RandomGenerator As New Random
   lblOutcome.Text = ""
   For i As Integer = 0 To 7
       intCountArray(i) = 0
   Next
   intNumber = Convert.ToInt32(txtNumber.Text)
   For k As Integer = 1 To intNumber
       intRandomNumber = RandomGenerator.Next(0, 8)
       intCountArray(intRandomNumber) = _
                     intCountArray(intRandomNumber) + 1
   Next
   For m As Integer = 0 To 7
       lblOutcome.Text = lblOutcome.Text & "The integer " & _
                 m.ToString & " appeared " & intCountArray(m) & _
                 " times" & ControlChars.NewLine
   Next
End Sub

Private Sub btnClear_Click(. . .) Handles btnClear.Click
   txtNumber.Text = ""
   lblOutcome.Text = ""
   txtNumber.Focus()
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
   Close()
End Sub
End Class
```

Examples of output:





QUESTION 5

Do the planning and write a program to solve the following problem:

Let the user enter 2 odd integers to serve as the range for obtaining random numbers.

Calculate the number in the middle of this range. Now, let the program obtain 7 random numbers and then display which one of these numbers is the nearest to the middle number.

	Description	Type	Name of variable
Input:	minimum value of range	integer	minValue
	maximum value of range	integer	maxValue
Output:	Number nearest to middle	integer	nearest

Algorithm:

MiddleNumber

```
        display "Provide 2 odd integers, the first lower than the second"
        enter minValue, maxValue
        middle = minValue + (maxValue – minValue) / 2
        distance = middle – minValue        ~  this number represents the biggest distance
        nearest = minValue                  ~  a value to start

        for j = 1 to 7
                randNumber = random(minValue, maxValue+1)
                randDistance = middle – randNumber
                randDistance = Abs(randDistance)     ~  use the absolute method
                if distance > randDistance
                        distance = randDistance
                        nearest = randNumber
                endif
        endfor
        display "The middle number is ", middle
        display "The random number nearest to the middle of the range is ", nearest
end
```

VB.NET SOLUTION

Take note that the program includes validation to test if the numbers entered are indeed off and if the second number is greater than the first number. The program will also display all numbers randomly generated on a list box, just for the students to see that the answer displayed is indeed correct.

Code in the code editor:

```
Public Class MiddleNumber

Private Sub btnStart_Click(. . .) Handles btnStart.Click

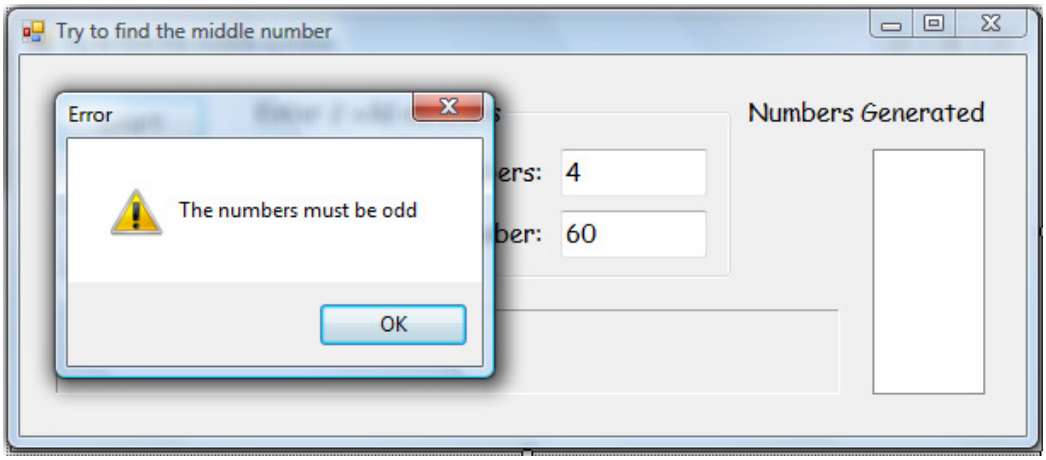
   Dim RandomGenerator As New Random
   Dim intRandNumber As Integer
   Dim intMinValue, intMaxValue As Integer
   Dim intMiddle As Integer
   Dim intDistance, intRandDistance As Integer
   Dim intNearest As Integer
   Dim blnValidMinOdd, blnValidMaxOdd As Integer
   blnValidMinOdd = Integer.TryParse(txtMinOdd.Text, intMinValue)
   blnValidMaxOdd = Integer.TryParse(txtMaxOdd.Text, intMaxValue)
```

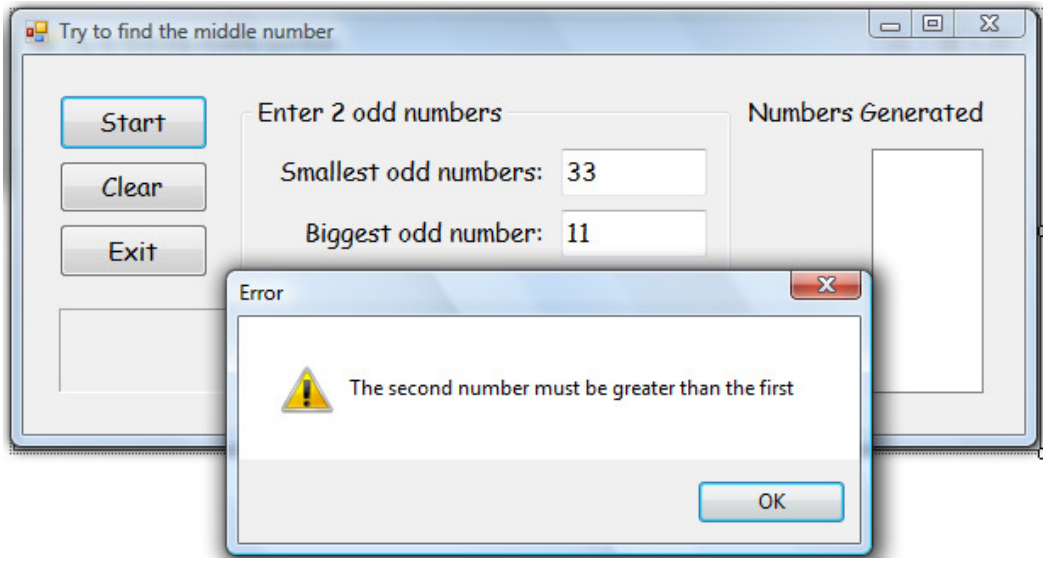
```
   If blnValidMinOdd AndAlso blnValidMaxOdd Then
    If intMinValue Mod 2 = 1 AndAlso intMaxValue Mod 2 = 1 Then
      If intMaxValue > intMinValue Then
         intMiddle = intMinValue + (intMaxValue - intMinValue) / 2
         intDistance = intMiddle - intMinValue
         intNearest = intMinValue
         For j As Integer = 1 To 7
           intRandNumber = _
           RandomGenerator.Next(intMinValue, intMaxValue + 1)
           lstNumbers.Items.Add(intRandNumber)
           intRandDistance = intMiddle - intRandNumber
           intRandDistance = Math.Abs(intRandDistance)
           If intDistance > intRandDistance Then
              intDistance = intRandDistance
              intNearest = intRandNumber
           End If
         Next
         lblAnswer.Text = "The middle is " & intMiddle.ToString _
         & ControlChars.NewLine & _
         "The random number nearest to the middle of the range is " _
         & intNearest.ToString
      Else
         MessageBox.Show(_
         "The second number must be greater than the first"_
         "Error", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
         txtMinOdd.Text = ""
         txtMaxOdd.Text = ""
         txtMinOdd.Focus()
      End If
    Else
      MessageBox.Show("The numbers must be odd", "Error", _
      MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
      txtMinOdd.Text = ""
      txtMaxOdd.Text = ""
      txtMinOdd.Focus()
    End If
  Else
    MessageBox.Show("The numbers must be integers", "Error",_
    MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    txtMinOdd.Text = ""
    txtMaxOdd.Text = ""
    txtMinOdd.Focus()
  End If
End Sub

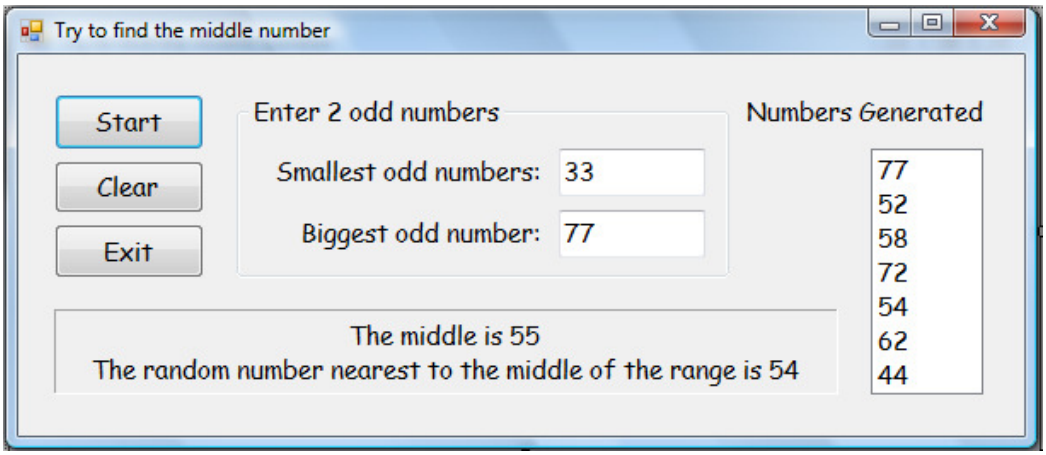
Private Sub btnClear_Click(. . .) Handles btnClear.Click
    txtMinOdd.Text = ""
    txtMaxOdd.Text = ""
    lblAnswer.Text = ""
    lstNumbers.Items.Clear()
    txtMinOdd.Focus()
End Sub

Private Sub btnExit_Click(. . .) Handles btnExit.Click
    Close()
End Sub
End Class
```

Examples of output:







QUESTION 7

Do the planning and write a VB.NET program to solve the following problem:

George wants to buy a nice big quad, but he needs a deposit of 60% of the price of the quad before the shop will sell it to him. He has to enter the price into your program and then you will have to determine how much his deposit will be and also how much the monthly instalments will be if he pays the remainder over a period of 2 years. He will have to pay 14.8% interest per annum on the outstanding amount. Display the instalment amount on the screen.

	Description	Type	Name of variable
Input:	price of quad	float	quadPrice
Output:	instalment on remainder	float	quadInstalment

Algorithm:

QuadOwner

display "Provide the price of the quad"
enter quadPrice
deposit = quadPrice * 0.6
remainder = quadPrice – deposit
quadInstalment = Pmt(0.148/12, 2*12, remainder)
display "The deposit amount is R", deposit
display "George must pay a monthly instalment of R", quadInstalment

end

VB NET SOLUTION:

In this example, the labels will be cleared when the user enters a new quad price.

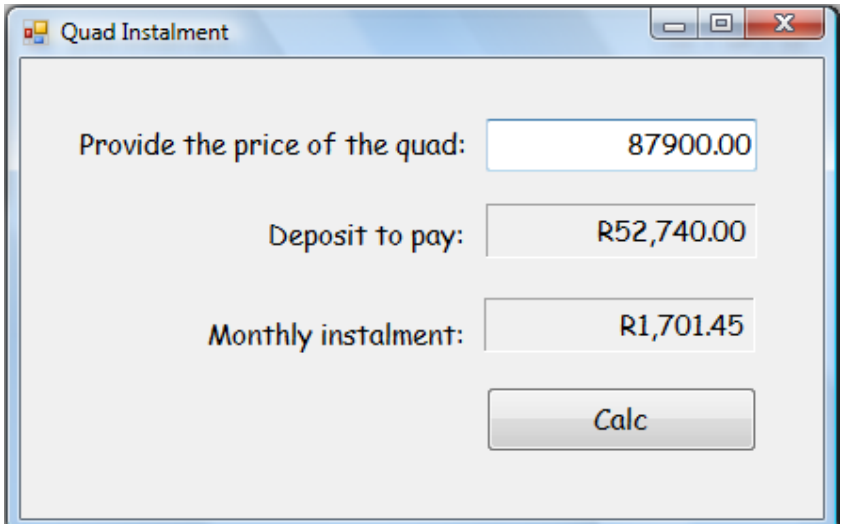
Code in the code editor:

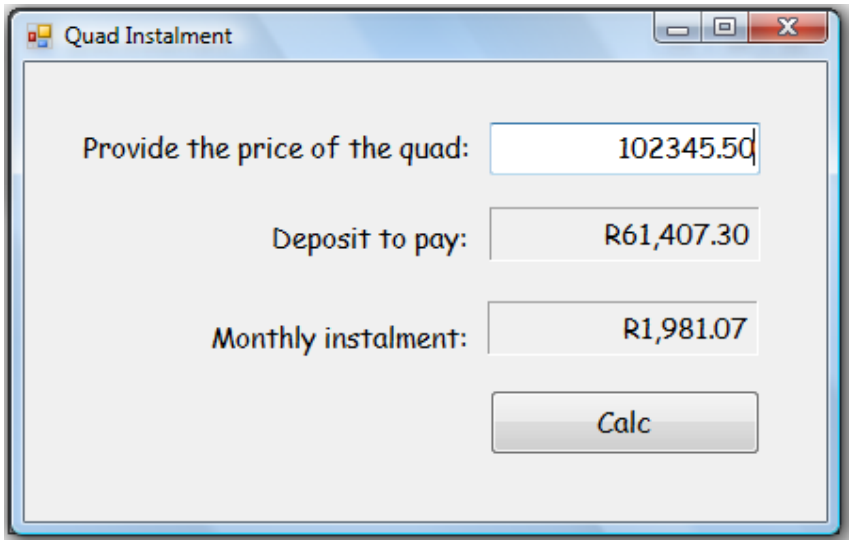
```
Public Class QuadOwner

Private Sub btnCalculate_Click(. . .) Handles btnCalculate.Click
   Dim decQuadPrice As Decimal
   Dim decDeposit As Decimal
   Dim decRemainder As Decimal
   Dim decQuadInstalment As Decimal
   Decimal.TryParse(txtQuadPrice.text, decQuadPrice)
   decDeposit = decQuadPrice * 0.6
   decRemainder = decQuadPrice - decDeposit
   decQuadInstalment = Financial.Pmt(0.148 / 12, 2 * 12, _
                       decRemainder)
   decQuadInstalment = Math.Abs(decQuadInstalment)
   lblDeposit.Text = "R" & decDeposit.ToString("N2")
   lblMonthlyInstalment.Text = "R" & decQuadInstalment.ToString("N2")
   txtQuadPrice.Focus()
End Sub
```

```
Private Sub txtQuadPrice_TextChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles txtQuadPrice.TextChanged
   lblDeposit.Text = ""
   lblMonthlyInstalment.Text = ""
End Sub
End Class
```

Examples of output:

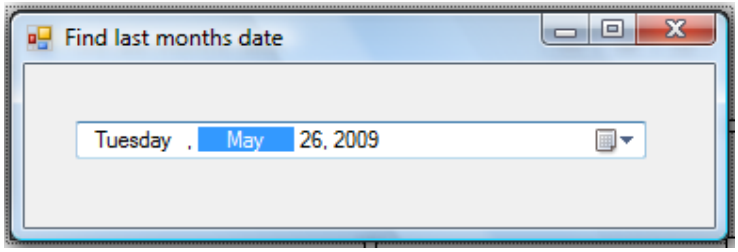




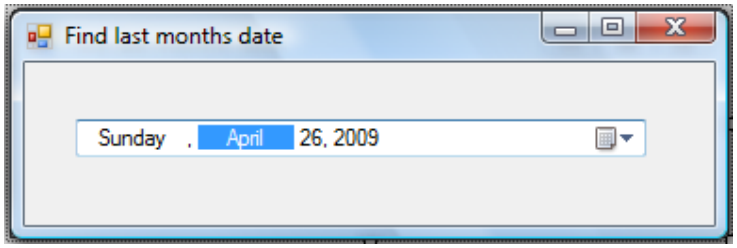
QUESTION 8

Write a VB.NET program and use the DateTimePicker to obtain and display today's date from the system. Use today's date to calculate and display the date of a month ago in short format. You may assume today is not in January.

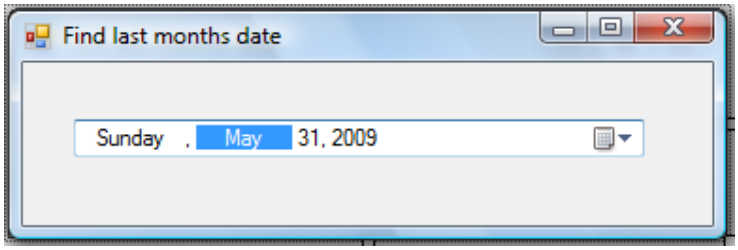
In a case like this, no logic is needed because the user can simply use the down arrow to obtain the previous month as indicated below. However, the programmer can use string functions in order to determine the date of a month ago and to display it on a label. You are welcome to try it out as an exercise.



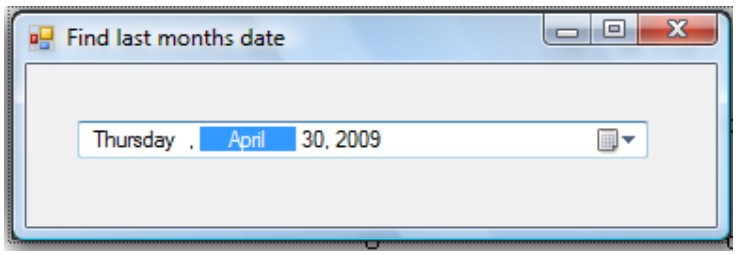
The month is already selected. Press the down arrow to obtain the date of one month ago.



We can repeat the exercise with a date of 31 May 2009.



If we now press the down arrow, it will not only adjust the month to April, but will automatically adjust the day to 30, because April does not have 31, but only 30 days.



CHAPTER 6
QUESTIONS & ANSWERS
COMPREHENSIVE EXAMPLES

QUESTIONS:

QUESTION 1

This program includes:

- A main procedure
- A sub procedure
- Value parameters
- Reference parameters
- Array manipulation
- Selection
- Calculations

The dean of the IT faculty of the EXCELLENT COLLEGE needs a program to determine the age distribution of all the students per gender, in the faculty. There are 500 students in the faculty. The input, the age and the gender (M or F) of the student, to this program must be done in a sub procedure. The main procedure will contain an array for the female students and another array for the male students.

The elements for both arrays will contain the following age groups:

Element number	Age group
0	16
1	17
2	18
3	19
4	20
5	21
6	22

A student may not be younger than 16 years. In such a case a suitable error message must be displayed and the age must be re-entered. If the input age is not in one of the groups, a male / female total for other ages must be incremented. (I.e. a student older than 22)

The output must be the following:

- Two lists of all the male and female totals per age group.
- The two other-ages totals.
- Total number of male and total number of female students
- The percentage of the other-ages per gender of the total number of students per gender. (Calculate and display in a sub procedure).

To test the program, you may use a group of 30 students.

QUESTION 2

This program includes:

- A main procedure
- Calculations
- Validation
- Select case statement
- Rounding

Solve this problem by planning a solution:

Employees of the HARD-WORKING COMPANY can earn their wages in four different ways:

	Code:	Input:
Monthly salary	S	salary
Commission	C	sales, percentage of sales
Hourly	H	number of hours, rate per hour
Job	J	number of jobs, rate per job

Enter the personnel number of the employee and then use a Switch statement to determine the type of wage the employee earns. Validate the input code and display a message if an incorrect code has been entered. Calculate the income and display the personnel number and income of every employee. The company has an unknown number of employees.

The average income for every group must also be displayed.

ANSWERS:

QUESTION 1

The dean of the IT faculty of the EXCELLENT COLLEGE needs a program to determine the age distribution of all the students per gender, in the faculty. There are 500 students in the faculty. The input, the age and the gender (M or F) of the student, to this program must be done in a sub procedure. The main procedure will contain an array for the female students and another array for the male students.

The elements for both arrays will contain the following age groups:

Element number	Age group
0	16
1	17
2	18
3	19
4	20
5	21
6	22

A student may not be younger than 16 years. In such a case a suitable error message must be displayed and the age must be re-entered. If the input age is not in one of the groups, a male / female total for other ages must be incremented. (I.e. a student older than 22)

The output must be the following:

- Two lists of all the male and female totals per age group.
- The two other-ages totals.
- Total number of male and total number of female students
- The percentage of the other-ages per gender of the total number of students per gender. (Calculate and display in a sub procedure).

To test the program, you may use a group of 30 students.

Planning:

	Description	Type	Name of variable
Sub Procedure (InputGenderAge):			
Parameters:	gender of student	character	refGender
	age of student	integer	refAge
Sub Procedure (OtherAge):			
Parameters:	male students other age	integer	valMaleOther
	female students other age	integer	valFemaleOther
	total male students	integer	valTotMale
	total female students	integer	valTotFemale
Output:	Percentage males other age	real	percMale
	Percentage females other age	real	percFemale
Main procedure (AgeDistribution):			
Has 2 arrays:	Array for males	integer	maleArray
	Array for females	integer	femaleArray
Output:	Contents of male array		
	Contents of female array		
	Male st other age group	integer	maleOtherAge
	Female st other age group	integer	femaleOtherAge
	Total number of males	integer	totalMale
	Total number of females	integer	totalFemale

Algorithm:

```
Sub Procedure InputGenderAge(refGender, refAge)          ~  input values
        Do
                display "Enter the gender of the student, M or F"
                enter refGender
                if refGender = "M" or refGender = "m" or
                   refGender = "F"  or refGender = "f"
                        ValidGender = True
                Else
                        ValidGender = False
        Loop Until ValidGender = True
        Do
                display "Enter the age of the student"
                enter refAge
                if refAge is numeric
                        if refAge >= 16
                                ValidAge = True
                        Else
                                ValidAge = False
                                Display "The student is too young to be a student"
                        Endif
                Else
                        ValidAge = False
                        Display "the age must be numeric"
                Endif
        Loop until ValidAge
endSub

Sub Procedure OtherAge (valMaleOther, valTotMale, valFemaleOther, valTotFemale)
        percMale = valMaleOther / valTotMale * 100
        percFemale = valFemaleOther / valTotFemale * 100
        display Round(percMale,1), "% of the males are other ages"
        display Round(percFemale,1) "% of the females are other ages"
endSub

Main Procedure AgeDistribution

        for k = 0 to 6                          ~initialize arrays
                maleArray(k) = 0
                femaleArray(k) = 0
        next

        maleOtherAge = 0
        femaleOtherAge = 0
        totalMale = 0
        totalFemale = 0
```

```
        for w = 1 to 500
                call InputGenderAge(gender, age)
                if gender = "M"
                        totalMale = totalMale + 1
                        if age > 15 AND age < 23
                                maleArray(age – 16) = maleArray(age – 16) + 1
                        else
                                maleOtherAge = maleOtherAge + 1
                        endif
                else
                        totalFemale = totalFemale + 1
                        if age > 15 AND age < 23
                                femaleArray(age–16) = femaleArray(age–16) + 1
                        else
                                femaleOtherAge = femaleOtherAge + 1
                        endif
                endif
        next

        display  "Male students"                        ~  display male groups
        for m = 1 to 7
                display "Age group ", 16+m, ":    ",maleArray(m)
        next
        display "Total male students:  ", totalMale

        display  "Female students"                      ~  display female groups
        for n = 1 to 7
                display "Age group ", 16+m, ":    ",femaleArray(n)
        next
        display "Total female students:  ", totalFemale

        display "Other male students totals;  ", maleOtherAge
        display "Other female students totals;  ", femaleOtherAge

        call otherAge (maleOtherAge, totalMale, femaleOtherAge, totalFemale)
end
```

VB.NET SOLUTION:

CODE IN THE CODE EDITOR:

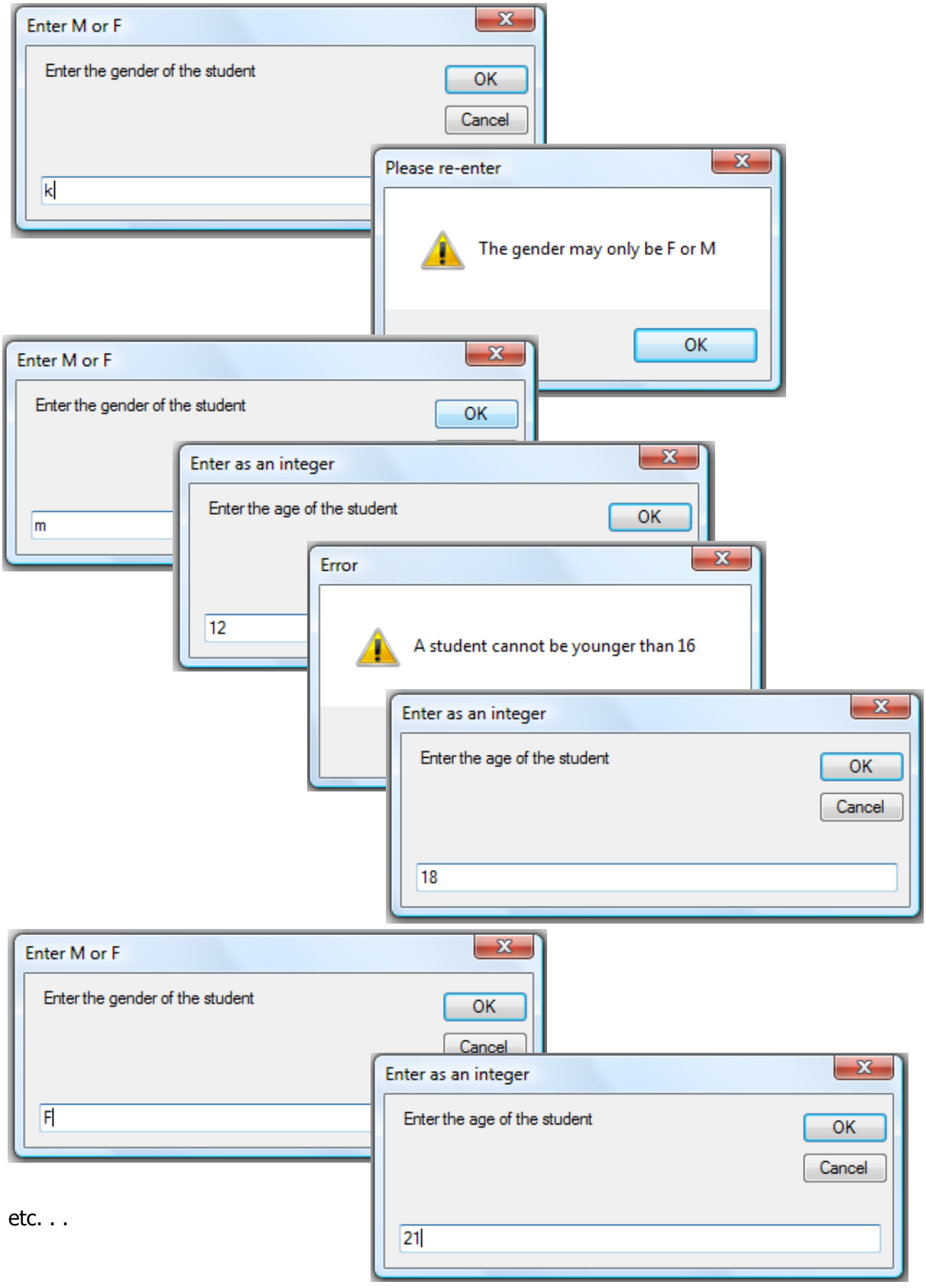
```
Public Class AgeDistribution

Private Sub InputGender(ByRef strGender As String, _
                        ByRef intAge As Integer)
   Dim blnAgeValid, blnGenderValid As Boolean
   Dim strAge As String
   Do
     strGender = InputBox("Enter the gender of the student", _
                          "Enter M or F")
     If strGender.ToUpper = "M" Or strGender.ToUpper = "F" Then
        strGender = strGender.ToUpper
        blnGenderValid = True
     Else
        blnGenderValid = False
        MessageBox.Show("The gender may only be F or M", _
                        "Please re-enter", MessageBoxButtons.OK, _
                        MessageBoxIcon.Exclamation)
     End If
  Loop Until blnGenderValid
  Do
    strAge = InputBox("Enter the age of the student", _
                      "Enter as an integer")
    blnAgeValid = Integer.TryParse(strAge, intAge)
    If blnAgeValid Then
       If intAge >= 16 Then
          blnAgeValid = True
       Else
          blnAgeValid = False
          MessageBox.Show("A student cannot be younger than 16", _
                          "Error", MessageBoxButtons.OK, _
                          MessageBoxIcon.Exclamation)
       End If
    Else
       blnAgeValid = False
       MessageBox.Show("The age must be an integer", "Error", _
                  MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
    End If
  Loop Until blnAgeValid
End Sub

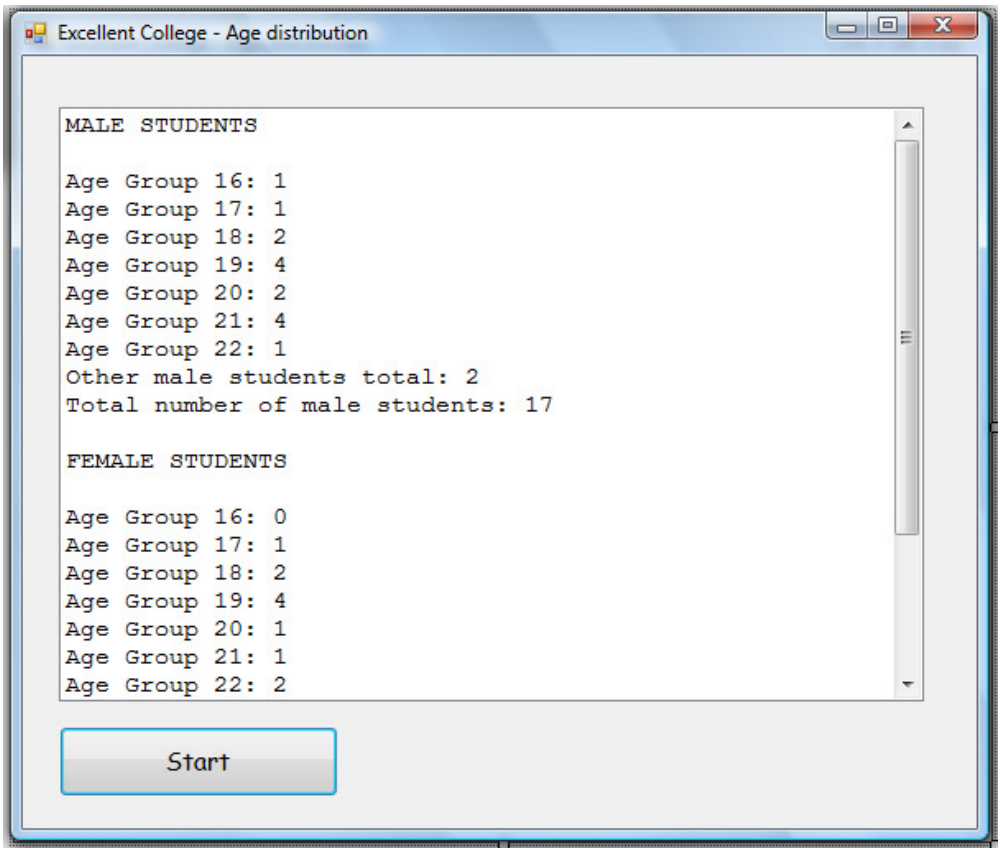
Private Sub OtherAge(ByVal intMaleOther As Integer, _
                     ByVal intTotMale As Integer, _
                     ByVal intFemaleOther As Integer, _
                     ByVal intTotFemale As Integer)
   Dim dblPercMale, dblPercFemale As Double
   dblPercMale = Convert.ToDouble(intMaleOther / intTotMale * 100)
   dblPercFemale = Convert.ToDouble_
                   (intFemaleOther / intTotFemale * 100)
   lstStudents.Items.Add("Percentage males of other ages: " & _
                    dblPercMale.ToString("N1") & "%")
   lstStudents.Items.Add("Percentage females of other ages: " & _
                    dblPercFemale.ToString("N1") & "%")
End Sub
```

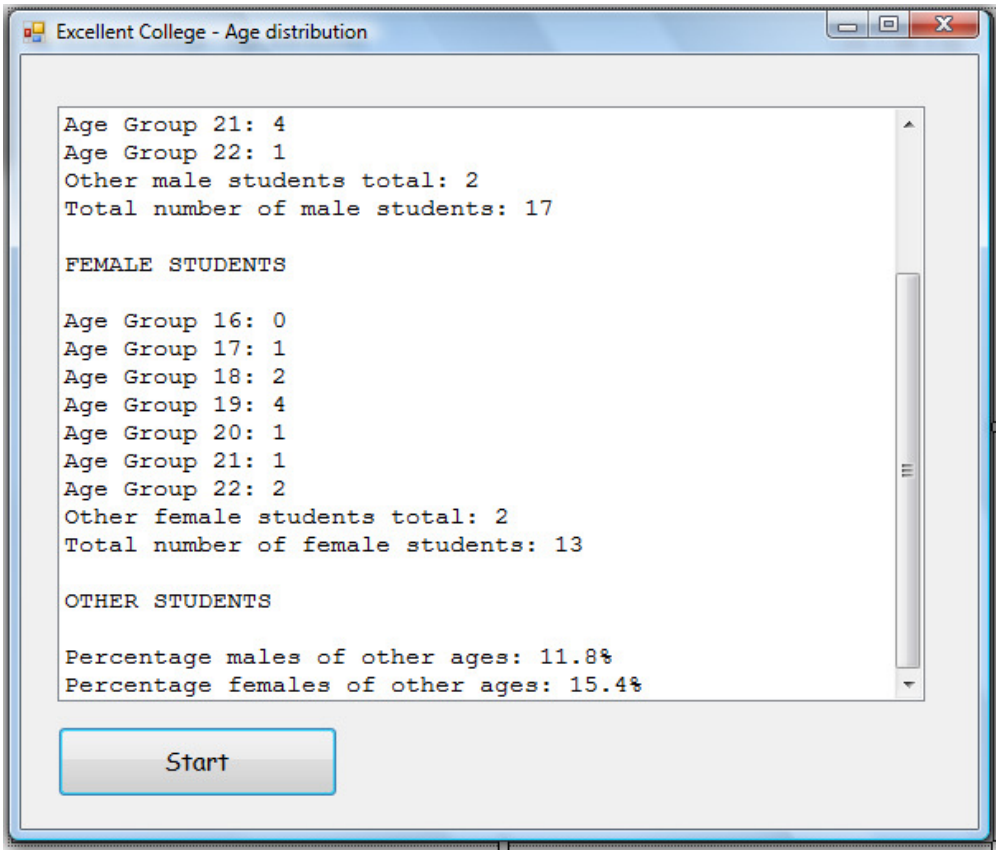
```
Private Sub btnStart_Click(. . .) Handles btnStart.Click
   Dim intMaleArray(6) As Integer
   Dim intFemaleArray(6) As Integer
   Dim intMaleOtherAge As Integer = 0
   Dim intFemaleOtherAge As Integer = 0
   Dim intTotalMale As Integer = 0
   Dim intTotalFemale As Integer = 0
   Dim strGender As String
   Dim intAge As Integer
   For k As Integer = 0 To 6
       intMaleArray(k) = 0
       intFemaleArray(k) = 0
   Next
   For w As Integer = 1 To 30
       Call InputGender(strGender, intAge)
       If strGender = "M" Then
          intTotalMale += 1
          If intAge > 15 AndAlso intAge < 23 Then
             intMaleArray(intAge - 16) += 1
          Else
             intMaleOtherAge += 1
          End If
       Else
          intTotalFemale += 1
          If intAge > 15 AndAlso intAge < 23 Then
             intFemaleArray(intAge - 16) += 1
          Else
             intFemaleOtherAge += 1
          End If
       End If
   Next
   lstStudents.Items.Add("MALE STUDENTS")
   lstStudents.Items.Add(" ")
   For m As Integer = 0 To 6
       lstStudents.Items.Add("Age Group " & (16 + m).ToString & _
                             ": " intMaleArray(m))
   Next
   lstStudents.Items.Add("Other male students total: " & _
                         intMaleOtherAge.ToString)
   lstStudents.Items.Add("Total number of male students: " & _
                         intTotalMale.ToString)
   lstStudents.Items.Add(" ")
   lstStudents.Items.Add("FEMALE STUDENTS")
   lstStudents.Items.Add(" ")
   For m As Integer = 0 To 6
       lstStudents.Items.Add("Age Group " & (16 + m).ToString & _
                             ": " & intFemaleArray(m))
   Next
   lstStudents.Items.Add("Other female students total: " & _
                         intFemaleOtherAge.ToString)
   lstStudents.Items.Add("Total number of female students: " & _
                         intTotalFemale.ToString)
   lstStudents.Items.Add(" ")
   lstStudents.Items.Add("OTHER STUDENTS")
   lstStudents.Items.Add(" ")
   Call OtherAge(intMaleOtherAge, intTotalMale, _
                 intFemaleOtherAge, intTotalFemale)
End Sub
End Class
```

Examples of output:



etc. . .


Excellent College - Age distribution
MALE STUDENTS
Age Group 16: 1
Age Group 17: 1
Age Group 18: 2
Age Group 19: 4
Age Group 20: 2
Age Group 21: 4
Age Group 22: 1
Other male students total: 2
Total number of male students: 17
FEMALE STUDENTS
Age Group 16: 0
Age Group 17: 1
Age Group 18: 2
Age Group 19: 4
Age Group 20: 1
Age Group 21: 1
Age Group 22: 2
Start


Excellent College - Age distribution
Age Group 21: 4
Age Group 22: 1
Other male students total: 2
Total number of male students: 17
FEMALE STUDENTS
Age Group 16: 0
Age Group 17: 1
Age Group 18: 2
Age Group 19: 4
Age Group 20: 1
Age Group 21: 1
Age Group 22: 2
Other female students total: 2
Total number of female students: 13
OTHER STUDENTS
Percentage males of other ages: 11.8%
Percentage females of other ages: 15.4%
Start

QUESTION 2

Solve this problem by planning a solution:

Employees of the HARD-WORKING COMPANY can earn their wages in four different ways:

	Code:	Input:
Monthly salary	S	salary
Commission	C	sales, percentage of sales
Hourly	H	number of hours, rate per hour
Job	J	number of jobs, rate per job

Enter the personnel number of the employee and then use a Select statement to determine the type of wage the employee earns. Validate the input code and display a message if an incorrect code has been entered. Calculate the income and display the personnel number and income of every employee. The company has an unknown number of employees.

The average income for every group must also be displayed.

Planning:

CalculateIncome

	Description	Type	Name of variable
Input:	personnel number	string	persNumber
	income code	alphabetic	code
	salary	real	salary
	sales	real	sales
	percentage of sales	real	perc
	number of hours	integer	noHours
	rate per hour	real	rateHour
	number of jobs	integer	noJobs
	rate per job	real	rateJob
Output:	income	real	income
	average salary group	real	aveSal
	average commission group	real	aveCom
	average hourly group	real	aveHour
	average job group	real	aveJob

Algorithm:

CalculateIncome

```
totSalary = 0          ~    initialise totals
totComm = 0
totHourPay = 0
totJobPay = 0
countSal = 0
countComm = 0
countHour = 0
countJob = 0
```

```
display "Enter the personnel number – X to indicate end of input"
enter persNumber

do while (persNumber <> X)
        display "enter income code"
        enter code

        select case code
                case "S"
                        display "enter the salary"
                        enter income
                        totSalary = totSalary + income
                        countSal = countSal + 1
                case "C"
                        display "enter sales"
                        enter sales
                        display "enter commission percentage"
                        enter perc
                        income = sales * perc / 100
                        totComm = totComm + income
                        countComm = countComm + 1
                case "H"
                        display "enter number of hours"
                        enter noHours
                        display "rate per hour"
                        enter rateHour
                        income = noHours * rateHour
                        totHourPay = totHourPay + income
                        countHour = countHour + 1
                case "J"
                        display "enter the number of jobs"
                        enter noJobs
                        display "enter the rate per job"
                        enter rateJob
                        income = noJobs * rateJob
                        totJobPay = totJobPay + income
                        countJob = countJob + 1
                case else
                        display code " entered – not correct!"
        end select

        display "Enter the personnel number – X to indicate end of input"
        enter persNumber
loop

if countSal <> 0
        aveSal =  Round(totSalary / countSal,2)
else
        display "No employee earns a salary"
endif
if countComm <>
        aveComm = Round(totComm / countComm,)
```

```
        else
                display "No employee earns commission"
        endif
        if countHour <> 0
                aveHour  =  Round(totHourPay / countHour)
        else
                display "No employee works per hour"
        endif
        if countJob <> 0
                aveJob  =  Round(totJobPay / countJob)
        else
                display "No employee does jobs"
        endif
end
```

CHAPTER 7
QUESTIONS & ANSWERS
Error Handling and Debugging Techniques

QUESTIONS:

QUESTION 1

Write the algorithm to solve the following problem and then do a trace table to test the correctness of the algorithm:

Tamara has quite a number of 10c, 20c, 50c R1 and R2 coins in her purse. She wants to know the monetary value of these coins. Calculate and display the amount.

Test the program using the following data

	10c	**20c**	**50c**	**R1**	**R2**
No coins:	12	4	0	8	14

QUESTION 2

Write the algorithm to solve the following problem and then do a trace table to test the correctness of the algorithm:

The GET-A-LIFE COLLEGE offers 3 different courses. Every course has 12 subjects, but a student may only register for a minimum of 2 and a maximum of 5 subjects per semester. All the subjects of one course have the same price but differ from subjects in other courses.

Course code	Price per subject
AB234	950.75
JK543	1150.00
MN789	1034.50

The user must enter the course code and the number of subjects. Calculate and display the total amount due by the student. If the student entered an incorrect course code, a suitable message must be displayed and the amount due will be zero. If the incorrect number of subjects were entered, the amount must also be zero and an error message must be displayed.

Test your algorithm using the following 3 sets of data:

Course:	JK543	No of subjects:	4
	MN789		6
	ZX987		2

QUESTION 3

Bongani sells tickets for a raffle. His clients pay less than the normal price of R5 per ticket if they buy more than one ticket according to the following table:

Number of tickets	Price per ticket
1 – 5	R5.00
6 – 12	R4.65
13 and more	R4.40

He wrote the following algorithm to solve the problem:

```
BuyTickets
        display "How many tickets do you want to buy?"
        enter noTickets
        ~       calculate the amount due
        if noTickets >= 1
                amount = noTickets * 5
        else
                if noTickets >= 6
                        amount = noTickets * 4.65
                else
                        amount = noTickets * 4.4
                endif
        endif
        display "The amount due is R", amount
end
```

If you run this program or do a trace table you will find that your answers are not correct.
How are you going to change the program to obtain correct results?
Correct the statements and test the program.

QUESTION 4

The following program ends abnormally and the programmer is advised to use display statements to have a look at the contents of the variables in order to find the error:

```
DoCalculations
        display "enter a value for A"
        enter A
        display "enter a value for B"
        enter B
        answerOne = A + B
        answerTwo = A / B
        display "answerOne is ",answerOne, " and answerTwo is ", answerTwo
end
```

QUESTION 5

Write an algorithm in at least 2 phases using stubs to solve the following problem:

A dry cleaning company called SPOTLESS has agents to collect clothes at different depots in town. The agent is paid for the number of items he/she collects according to the following table:

Number of items	Payment per items
0 - 10	R3.50
11 - 24	R4.70
25 and more	R6.25

The input to this program is the name of the salesperson and the number of items. The program must terminate when a name = ZZZ is entered.
A function must receive the number of items and must calculate the agent's pay. The main procedure will display the income of every agent.
The total amount paid, that is accumulated in a sub procedure, must be displayed at the end of the program.

ANSWERS:

QUESTION 1

Algorithm:

```
DetermineAmount
        display "enter the number of 10c coins"
        enter coins
        amount = coins * 0.1            ~  calculate the value
        display "enter the number of 20c coins"
        enter coins
        amount = amount + coins * 0.2       ~ calculate the value and add to total
        display "enter the number of 50c coins"
        enter coins
        amount = amount + coins * 0.5
        display "enter the number of R1 coins"
        enter coins
        amount = amount + coins
        display "enter the number of R2 coins"
        enter coins
        amount = amount + coins * 2

        display "The amount is R", amount
end
```

Test the program using the following data

	10c	20c	50c	R1	R2
No coins:	12	4	0	8	14

Trace Table:

Instruction	No coins	amount	output
display			Enter number of 10c coins
enter	12		
calculate		1.20	
display			Enter number of 20c coins
enter	4		
calculate		80c	
display			Enter number of 50c coins
enter	0		
calculate		0	
display			Enter number of R1 coins
enter	8		
calculate		8.00	
display			Enter the number of R2 coins
enter	14		
calculate		28.00	
display			The amount is R38.00

The trace table can now be tested to ensure that our result is correct

	10c	20c	50c	R1	R2
No coins:	12	4	0	8	14
Value:	1.20	80c	0	R8	R28

When all the values in the last row are added, the result is R38.00 !!

QUESTION 2

Algorithm:

```
StudentFees
        for b = 1 to 3
                display "enter the student number"
                enter stNo
                display "enter the number of subjects
                enter subjects
                if subjects < 2 OR subjects > 5
                        display "error – number of subjects: ", subjects
                        fee = 0
                else
                        display "enter course code"
                        enter code
                        select case code
                                case "AB234"
                                        fee = subjects * 950.75
                                case "JK543"
                                        fee = subjects * 1150
                                case "MN789"
                                        fee = subjects * 1034.5
                                case else
                                        fee = 0
                                        display code, " incorrect course code"
                        end select
                endif
                display "The fee for student number ", stNo, " is R", fee
        next
end
```

Trace Table:

Instruction	b	stNo	Code	subjects	Fee	If outcome	Output
for	1						
display							Enter student number
enter		4321					
display							Enter no of subjects
enter				4			
If						False	
display							Enter course code
enter			JK543				
select						True	
calculate					4600.00		
display							The fee for student no 4321 is R4600.00
for	2						
display							Enter student number
enter		5432					
display							Enter no of subjects
enter				6			
If						True	
display							Error – number of subjects: 6
calculation					0		
display							The fee for student no 5432 is R0.00
for	3						
display							Enter student number
enter		9321					
display							Enter no of subjects
enter				2			
If						False	
display							Enter course code
enter			ZX987				
select						False	
calculation					0		
display							ZX987 incorrect course code
display							The fee for student number 9321 is R0.00

QUESTION 3

Bongani sells tickets for a raffle. His clients pay less than the normal price of R5 per ticket if they buy more than one ticket according to the following table:

Number of tickets	Price per ticket
1 – 5	R5.00
6 – 12	R4.65
13 and more	R4.40

Solution: You will find that the price of all the tickets are R5.00, because the first if statement will include all possibilities.

After correcting the if statements, the program will execute perfectly!

```
BuyTickets
        display "How many tickets do you want to buy?"
        enter noTickets
        ~       calculate the amount due
        if noTickets >= 13
                amount = noTickets * 4.4
        else
                if noTickets >= 6
                        amount = noTickets * 4.65
                else
                        amount = noTickets * 5
                endif
        endif
        display "The amount due is R", amount
end
```

QUESTION 4

The following program ends abnormally and the programmer is advised to use display statements to have a loop at the contents of the variables in order to find the error:

We will include a display statement before the first calculations that uses the variables A and B:

```
DoCalculations
        display "enter a value for A"
        enter A
        display "enter a value for B"
        enter B
        display "A = ",A,"    B = ",B
        answerOne = A + B
        answerTwo = A / B
        display "answerOne is ",answerOne, " and answerTwo is ", answerTwo
end
```

It is quite possible that we will find one of the following reasons for the abnormal ending of the program:

- A and/or B does not contain numerical value(s)
- B contains the number zero, and the program is trying to divide by zero
- The enter key was hit before anything was entered for A and B
- The program does not indicate that numerical values must be entered for A and B

QUESTION 5

There are many ways to write this program and to use stubs. This is only one of the possible answers:

Algorithm:

First Phase:

```
Function CalcPay (valNumber)

        pay = 100                                   ~  stub

return pay

Sub Procedure AccTotal (refTotal, valPay)
        refTotal = refTotal + valPay
End Sub


PayOfAgents                 ~  main procedure

        total = 0
        display "enter the name of the agent – ZZZ to terminate input"
        enter name
        do while name <> "ZZZ"
                display "enter the number of items"
                noItems = 10                        ~  stub
                pay = CalcPay (noItems)             ~  call function
                total = 1000                        ~  stub
                display name, " earns R", pay
                display "enter the name of the agent – ZZZ to terminate input"
                enter name
        loop
        display "The total amount of pay is R", total
end
```

Final Program:

```
Function CalcPay (valNumber)

        if valNumber <= 10
                pay = valNumber * 3.5
        else
                if valNumber <= 24
                        pay = valNumber * 4.7
                else
                        pay = valNumber * 6.25
                endif
        endif
return pay

Sub Procedure AccTotal (refTotal, valPay)
        refTotal = refTotal + valPay
End Sub


PayOfAgents                     ~  main procedure

        Total = 0
        display "enter the name of the agent – ZZZ to terminate input"
        enter name
        do while name <> "ZZZ"
                display "enter the number of items"
                enter noItems
                pay = CalcPay (noItems)                 ~  call function
                call AccTotal (total, pay)
                display name, " earns R", pay
                display "enter the name of the agent – ZZZ to terminate input"
                enter name
        loop
        display "The total amount of pay is R", total
end
```

Appendix A
GUI Guidelines

When designing a user interface, there are many guidelines to adhere to in order to make the form as user friendly as possible. Most of these guidelines have been discussed in level 2 and this appendix gives a summary of all the guidelines discussed in level 2 as well as guidelines for new controls used in level 3. All the graphical user interfaces designed for examples in the level 3 notes adhere to these guidelines.

General guidelines:

- Align the borders of all controls and try to make similar components exactly the same size (width and height).
- Graphics and color should be used sparingly. If you use a graphic, use a small one and place it at a location where it will not distract the user. Limit the number of colors to three.
- Use a font that is easily readable. Keep the font and the size consistent.
- The information on a form should flow either horizontally or vertically, with the most important information at the top left corner.
- Keep margins consistent.
- Related controls are typically placed close together on the form.

Guideline for labels:

- Label controls must be used to display information that you don't want the user to change while the application is running.
- A label to identify a text box must be placed to the left or above the text box, should end with a colon and should be typed in sentence capitalization.
- Labels to identify text boxes should be left aligned.
- Labels that identify controls should have their BorderStyle property set to none.
- Labels that display program output, such as the answer of a calculation, normally has the BorderStyle property set to FixedSingle or Normal3D.
- Restrict the highlight of text to headings.

Guidelines for text boxes:

- Although the background and foreground color of a text box can be set, the number of colors on an interface should be limited to 3. The colors you choose should complement one another and should not distract the attention of the user.
- Related text boxes should be grouped together. A group box may be used for this purpose.
- A text box must always contain a related label in order to give clear instructions of what should be entered in the text box
- Be consistent with fonts and don't use a different font for every text box on a form

Guidelines for group boxes:

- Use a group box to group related controls together.
- Maintain a consistent margin from the edge of the group box.
- Align the borders of the controls in the group box where possible to minimize the number of different margins within the group box.

Guidelines for buttons:

- A button control is used to perform an immediate action when clicked
- The text property of a button should be entered in book title capitalization. (Capitalize the first letter of every word)
- When buttons are positioned horizontally on the screen, all the buttons should be of the same height. It is preferred that the widths are also the same although they may vary.
- When buttons are positioned vertically on the screen, all buttons should be of the same height and width.
- The commonly used button should be placed first.
- This button that represents the user's most used action should also be the default button.
- When assigning an access key to a button use the first letter of the caption unless another letter provides a more meaningful association. If the first letter has already been used for another access key, use a distinctive consonant. Lastly use a vowel or a number.

Guidelines for picture boxes:

- Graphics and color should be used sparingly in an interface.
- If you use an image in the interface, use a small one and place it in a location that will not distract the user.

Guidelines for check boxes:

- Use a check box when you want the user to select one or more choices from a group of options
- A check box can also be used on its own if you want the user to indicate whether a certain condition is true or false e.g. indicate whether you are a member of the club – yes or no.
- The label in the text property of the check box should be entered in sentence capitalization.

Guidelines for radio buttons:

- Use radio buttons when you want to limit the user to one of two or more choices when only one of those choices may be selected.
- The minimum number of radio buttons in a group is two and the recommended maximum is seven.
- It is good programming practice to designate a default radio button is each group of radio buttons.
- Radio buttons can be grouped together by means of a group box.
- Only one radio button in each group can be selected at a given time.
- Every radio button must have a text entry to indicate the purpose of the radio button.
- Use sentence capitalization for the optional identifying label in a group box control

Appendix B
Brief Overview of Database Application Development and Concepts

Using Microsoft Visual Basic 2008 Express Edition 'wizards'

Notes compiled by: Johan Kirsten, L Roux, HG Erasmus, B Buitendag

Introduction

Welcome to the database application sections where we will discuss the exciting topic of building data-driven software applications using the build in Visual Studio.NET wizards. This section is only intended to provide an overview of DB applications in VB. A more detailed study of DB's and DB applications in VB.NET is discussed in the next level of your course.

A database application is a powerful business tool used for day-to-day operations or to project important business decisions by analyzing data stored within a database. During this chapter, we will discuss topics related to databases and software applications implementing the features of databases. Amongst these are the following:

- Database concepts and terminology common to database programming.
- Using Microsoft ADO.NET to connect to databases and utilize the data.
- Implementing error handling and flow control into a database application.
- Validating data in order to ensure data accuracy and integrity
- Using other programming techniques with data driven application development.

Data need to be organized in specific ways to be useful and to provide information for the user. When data is organized in a useful way, it can be used to obtain lots of results.

One way of organizing data is to plan, design and develop a database where data can be manipulated to provide the user with meaningful results.

To be able to provide these results, or knowledge a program is needed to prepare results and reports requested by the user.

A database consists of two or more tables that can reference one another in such a way that these tables do not both contain the same data, but only a link, a relationship between one another.

Another definition: A database can be described as a collection of data or facts regarding a specific topic, e.g. details of a CD collection or details concerning clients. A database contains tables of related facts or data, together with resources or objects that allow us to use the database.

Properties of a Tabular Relation

A single table has the following characteristics:

- No duplicate rows
- No duplicate column names

- Physical row order does not matter
- Physical column order does not matter
- Values are atomic (can not be decomposed further)

The components of a database

- **Data**

The data of a database is shared and integrated i.e. the data is shared amongst applications. Redundant data is not allowed in a database.

- **Users**

There are 3 classes of users
End Users
DBA's
Programmers

- **Hardware**

- **Software**
Database Management System (DBMS)
Application Software

The Functions of a Database

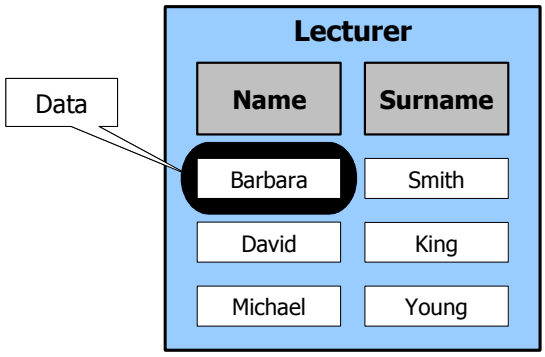
The data in a database can manipulated to enable the user to access and get the information required.

Basic operations:
- Create new tables
- Insert new data
- Update data
- Query data
- Delete data

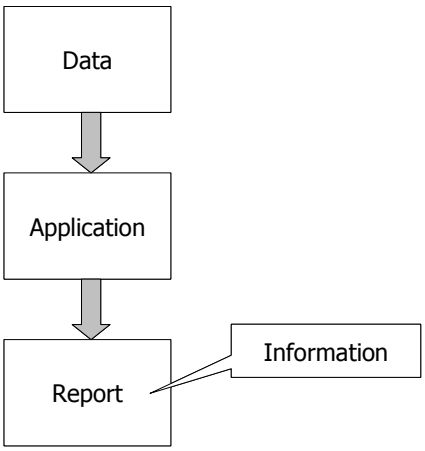
Database concepts

In order to successfully build data driven applications, it is very important to be acquainted with the basic terminology used in data driven applications. During this section, we will look at some terminology used in this chapter. It is very important to understand these terms as these are the basic building blocks for a successful database application developer. Please take the time to review the terminology and feel free to reference back to this section in the future.

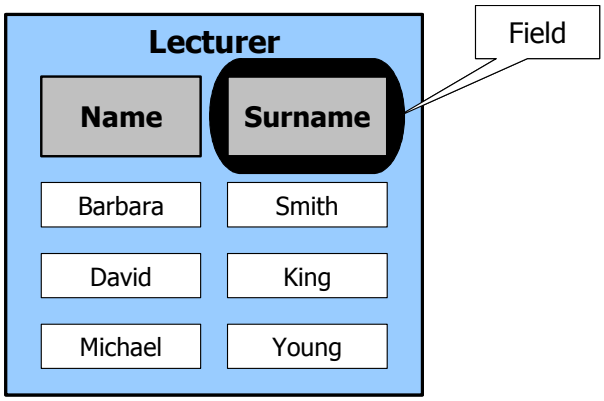
1) Data – are raw facts that do not mean anything by itself, but with the proper manipulation it is a valuable business asset. For instance, your name is a piece of data, but it does not hold any value by itself. If you combine your name and surname together, you are able to identify yourself properly to somebody else, giving those two pieces of data much more purpose than being on their own.



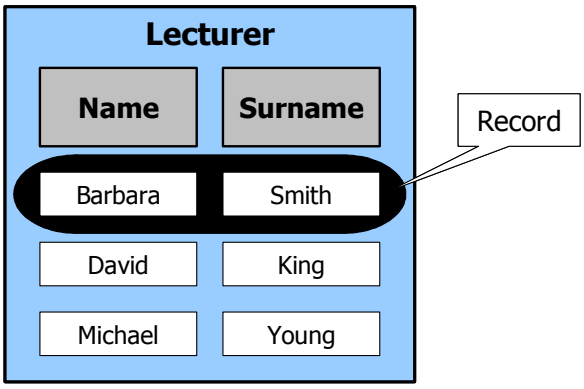
2) Information – is data in a processed form. Information is the strategic combination and manipulation of data so that it has value to the stakeholder in possession of the particular data. An example of information is the financial statements of a company. The content of the financial statements is derived from the data gathered throughout the year and thus forms a useful report which may be used to determine the financial stance of the company.



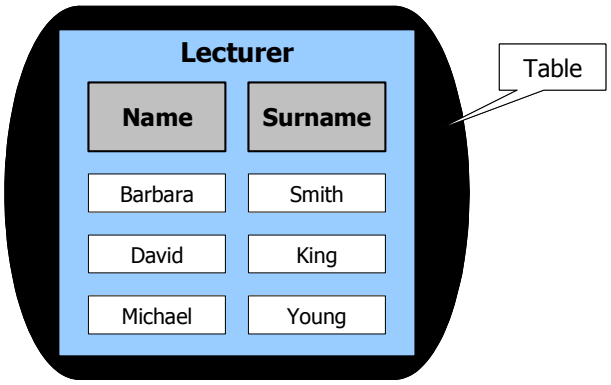
3) Field – within a database system, a piece of data is referred to as a field. For instance, the "Surname" is called a field and stores the data which should be something like the following, "Smith". In other words, a piece of data is labelled so that we may identify it, and we call this type of identification a field.



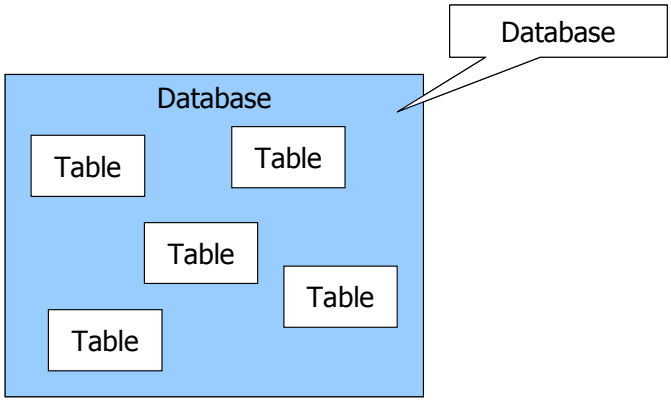
4) Record – is the combination of more than one field. By carefully combining certain fields, it is possible to describe an entity with the data available. For instance, if the record contains the "Name" and "Surname", a lecturer can easily be described and identified using the data.



5) Table – a table is a collection of related records. A list of lecturers' mutual pieces of data arranged into records can be grouped together to form a table.



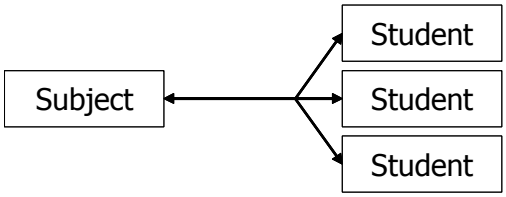
6) Database – a database is a collection of tables and various other parts of data driven functions, but for our purposes it is good enough to look at only the tables. A collection of tables describing related entities forms a database of raw facts that may be used to perform business processes.



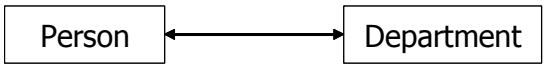
7) Database Management System – is a software system providing the functionality to manage the database content such as tables, record, fields and data stored within the database.

8) Database relationships – provides an association between two different, but related entities. For instance, a lecturer schools many students. This is an example of a database relationship. It provides a link between the "Lecturer" and "Student" entities and ensures data integrity between the two entities. For the purpose of this text, we will only look into two types of database relationships:

a. One-to-many (1 : M) – this type of database relationship, indicates that for every one entity on the left, there is more than one entity on the right. For instance, for every one "Subject" a group of "Students" may enrol.



b. One-to-one (1 : 1) – this type of database relationship, indicates that for every one entity on the left, there may only be one entity on the right. For instance, a department within the college may only have one head of department, and a person may only be head of department over one department at a time. This means that at any given time, a "Person" will be linked to only one "Department".



9) ERD – Entity Relationship Diagram – this diagram is used to draw the database components and the relationships between them in a graphical manner. Typically the ERD will contain tables, relationships between the tables, the definition or list of fields within the table and also the types of relationships between the different tables.

10) SQL – Structured Query Language – provides a manner to communicate in a standardized way to the contents within a database. SQL uses specific syntax and keywords to perform duties. Please refer to the appendix for a detailed walkthrough on understanding SQL and the usage thereof.

11) Class – is the collection of related properties or fields within a database table describing a specific group of related entities. The class provides behavioural functionality called methods or functions to manipulate data and perform specific actions. Please refer to the appendix for a detailed walkthrough on classes and the usage thereof.

Microsoft ADO.NET

Microsoft ADO.NET is a collection of classes provided with the Microsoft .NET Framework for software developers to build data driven software applications. ADO.NET consists of data access and manipulation classes. Data access classes provide the necessary functionality to connect to a data source and communicate with the data source. Data manipulation classes provide the functionality to utilize the data retrieved from the data source within a software application. Please study the following section to gain a better understanding on how Microsoft ADO.NET implements the aforementioned classes.

Data access

To access the data stored within a database file or database system, we need to connect to the database file or database system programmatically. Microsoft provides various options to connect to these types of database files or database systems. During this section, we will look at some of the options which are:

1. ODBC .NET Data Provider
2. OLE DB .NET Data Provider
3. Microsoft SQL Server .NET Data Provider

ODBC .NET Data Provider

The ODBC .NET Data Provider is an implementation of the ODBC standard developed in earlier years to connect to data sources such as Microsoft Access, Microsoft Excel and even plain text files using a standardized set of software libraries. ODBC is useful for connecting to many types of databases and is still in wide usage today.

OLE DB .NET Data Provider

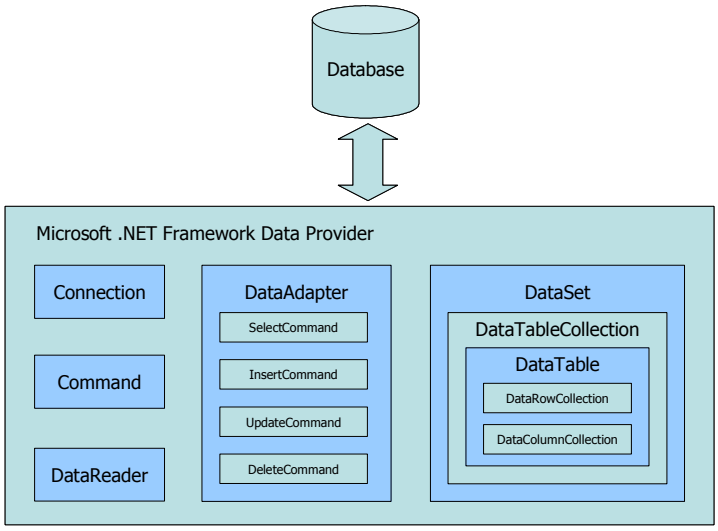
The OLE DB .NET Data Provider is an upgraded and improved version of the ODBC standard developed by Microsoft. This data provider harnesses the power and versatility of the Microsoft technologies and software libraries to build more efficient and better software applications. OLE DB provides better support for a wider range of data sources which are not supported or implemented in the ODBC standard.

Microsoft SQL Server .NET Data Provider

The Microsoft SQL Server .NET Data Provider is specially built for connecting to the Microsoft SQL Server systems. Microsoft SQL Server is an advanced database management system used for providing multiple access and management of the data contained within the database. The Microsoft SQL Server .NET Data Provider caters for the requirements of Microsoft SQL Server data sources exclusively.

Data manipulation

A data provider performs the duties of transferring data between the software application and the data source. To do something with the data, we need a component to retrieve the data from a data provider and do something with that data in order for the data to mean something, in other words for the data to become information. The Microsoft .NET Framework provides powerful and easy to use libraries called the ADO.NET data access and manipulation libraries. During the next section we will look at a diagram explaining how the Microsoft ADO.NET component works:



The above diagram demonstrates how the Microsoft ADO.NET classes fit together.

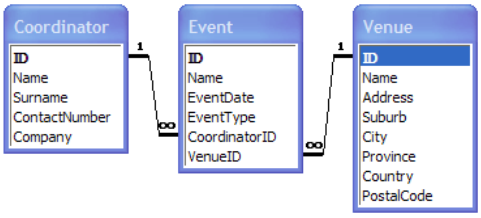
As part of the scope of this course it is only required to have knowledge of how Data connectivity is achieved in a VB.NET ADO application. The ADO framework allows for the connection to MSAccess databases.

Creating an ADO VB.NET Master-Detail application by utilizing the 'DB' wizards – Step by step example.

Study the following description of a DB scenario. The subsequent example utilizes Visual Studio.NET's build in 'wizard' capabilities to construct a database application interface that presents the Venue and Event tables in a master detail relationship.

Case Study: Fascination Productions & Events

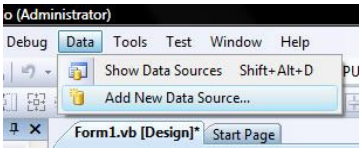
Fascination Production & Events contacted Software Solutions to build a software application to handle the scheduling of events. Fascination Production & Events is a dynamic company specializing in the organization of important and exciting events within the entertainment industry.
Amongst these events are road shows, live performances of bands, comedian events, sporting events, and theatre performances. The company employs coordinators who handle the organization of a specific event which is held at a specific venue.
Software Solutions' duty is to provide a software system that will keep track of the coordinators, venues, and events. Please study the relationships between the tables for the subsequent exercises at the end of the chapter.



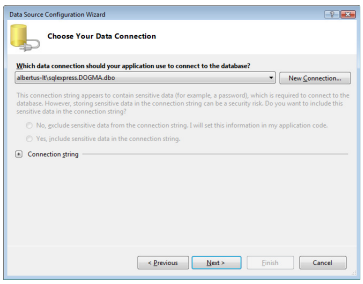
Step by step example

Create a standard Application and save the application with an appropriate name.

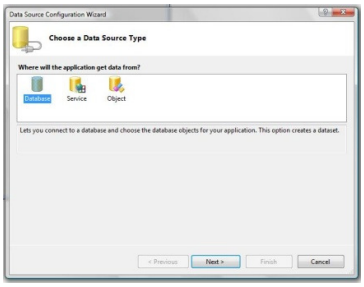
Click on the Data menu option and Select Add new Data Source



Click in the New Connection Button



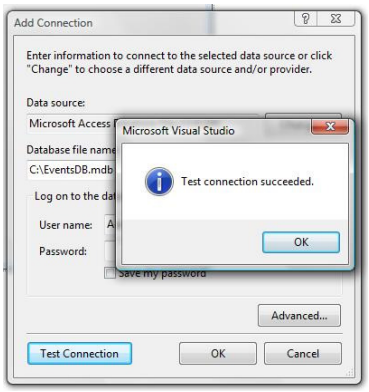
Select the Database option as a Data source and click on Next



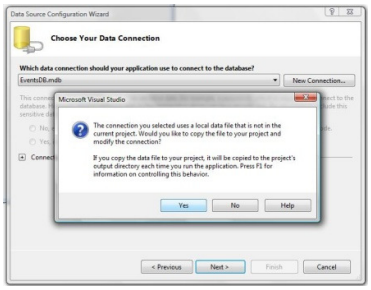
Select the Microsoft Access OleDB as a datasource and browse and select the Events.DB database (Access Database_



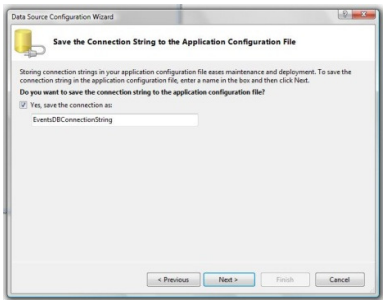
Click on the Test Connection Button to determine whether the a connection could be successfully established.



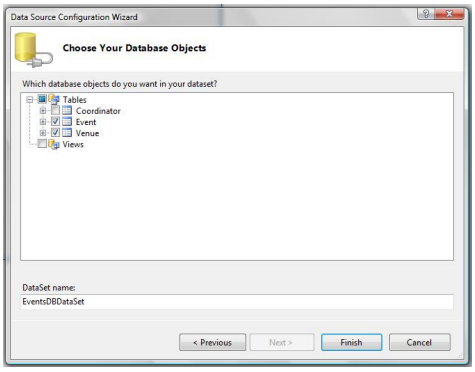
When prompted to include the selected DB (Which is EventsDB in our case) as part of the project click on Yes.



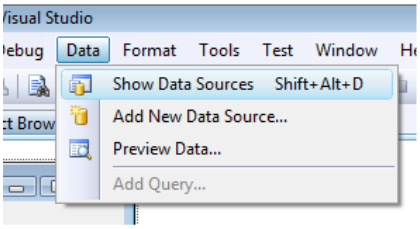
Save the connection string. (Supply one if not automatically supplied)



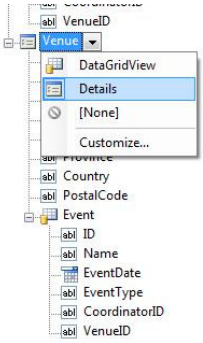
As our application is only to use and manipulate data from the Venues and Event tables only these are selected.



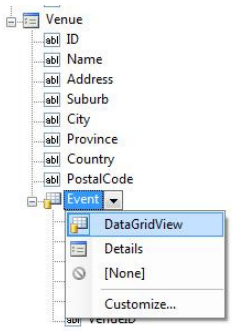
Select the Show Data Sources menu option.



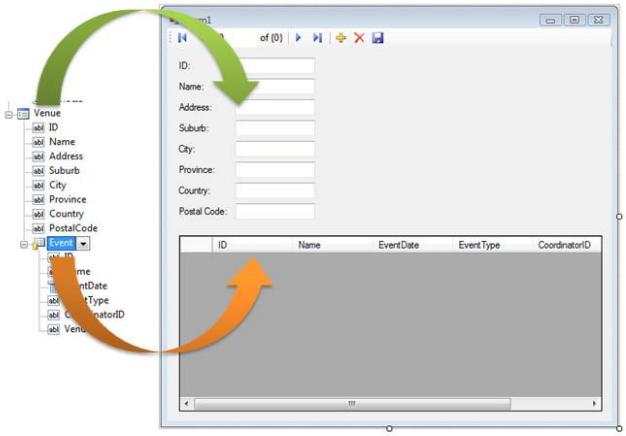
The Data Sources window provides the user with a detailed layout of the tables that was forms part of the DataSet of the application.
Select the Venue table and click on the details view



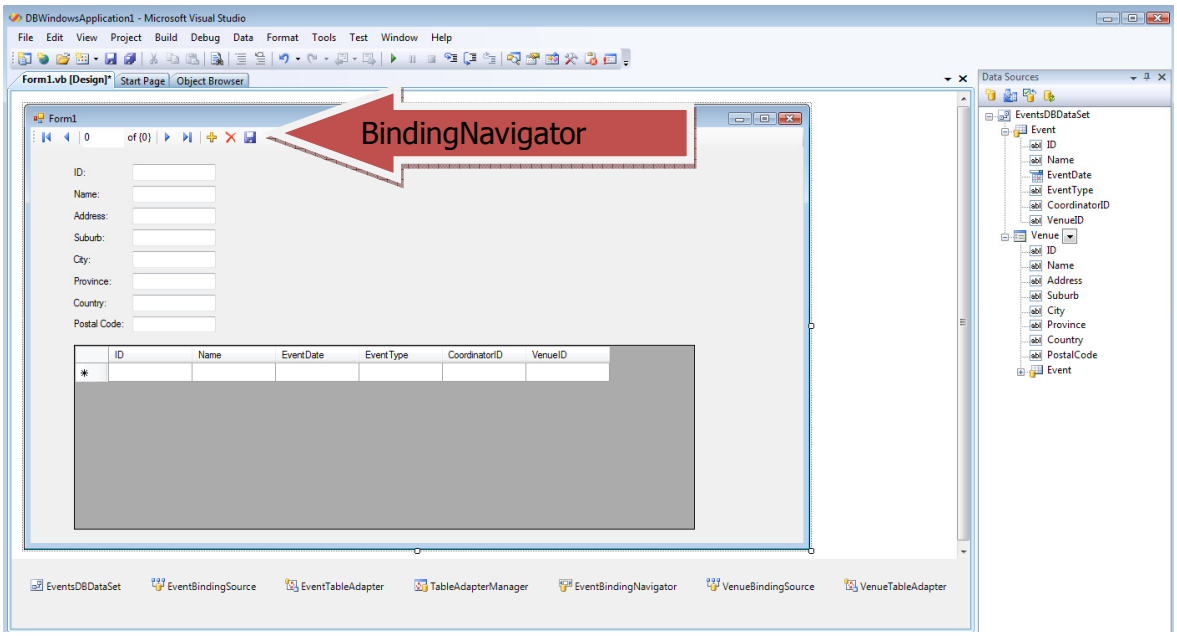
Thereafter select the events table and click on DataGridView



Drag the Venue and the Event tables to the form. Notice how the compiler automaticly generates the components. A DataGridView is used to present the detail records of the events. Take note that only the events held at a specific venue is shown.



After dragging the tables onto the form the compiler (via the wizard) automatically adds a set of components to your application.



A BindingNavigator connected to the Venues table is automatically inserted to the form. This component is used to navigate and supply instructions pertaining to the records of the Master Table (Venues)

Save compile and run the application. Notice how the detail records are chanced according to the Foreign and PK relationship settings as the user navigates through the records of the venue table.

