Workbook

O Level & IGCSE Computer Science

P2: Programming (VB.NET)

Inqilab Ruknuddin Patel

Computer Science with Inqilab Patel
+923002724734 /inqilabpatel inqilab patel
@inqilab inqilab-patel ruknuddin.com

2.2 Programming

Visual Basic .NET (VB.NET) is an object-oriented computer programming language implemented on the .NET Framework. Although it is an evolution of classic Visual Basic language, it is not backwards-compatible with VB6, and any code written in the old version does not compile under VB.NET. Like all other .NET languages, VB.NET has complete support for object-oriented concepts. Everything in VB.NET is an object, including all of the primitive types (Short, Integer, Long, String, Boolean, etc.) and user defined types, events, and even assemblies. All objects inherit from the base class Object.

VB.NET is implemented of Microsoft's .NET framework. Therefore it has full access all the libraries in the .Net Framework. It's also possible to run VB.NET programs on Mono, the open-source alternative to .NET, not only under Windows, but even Linux or Mac OSX.

The following reasons make VB.Net a widely used professional language:

- Modern, general purpose.
- Object oriented.
- Component oriented.
- Easy to learn.
- Structured language.
- It produces efficient programs.
- It can be compiled on a variety of computer platforms.
- Part of .Net Framework.
- Strong Programming Features VB.Net

VB.Net has numerous strong programming features that make it endearing to multitude of programmers worldwide. Let us mention some of these features:

- Boolean Conditions
- Automatic Garbage Collection
- Standard Library
- Assembly Versioning
- Properties and Events
- Delegates and Events Management
- Easy to use Generics
- Indexers
- Conditional Compilation
- Simple Multithreading

VB.NET – ENVIRONMENT

Integrated Development Environment (IDE) For VB.Net

Microsoft provides the following development tools for VB.Net programming:

- Visual Studio 2010 (VS)
- Visual Basic 2010 Express (VBE)
- Visual Web Developer

The last two are free. Using these tools you can write all kinds of VB.Net programs from simple command-line applications to more complex applications. Visual Basic Express and Visual Web Developer Express edition are trimmed down versions of Visual Studio and has the same look and feel. They retain most features of Visual Studio. In this tutorial, we have used Visual Basic 2010 Express and Visual Web Developer

Download and Setup

Download VB Express and install it on your computer at home.
http://www.microsoft.com/visualstudio/eng/products/visual-studio-2010-express

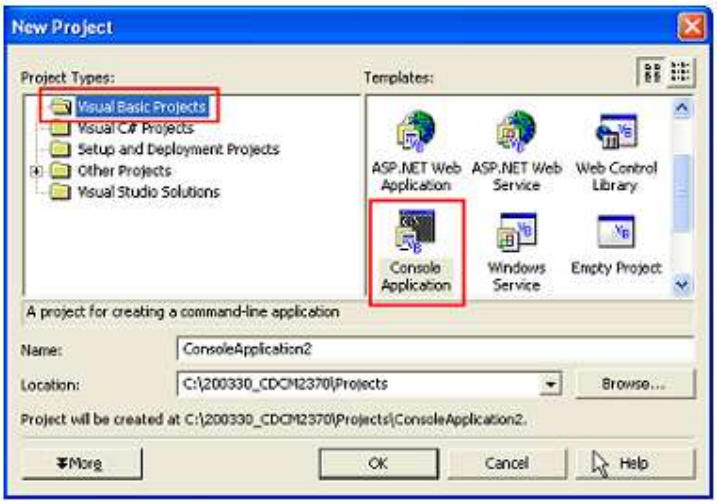
If you don't run Windows as your operating system you can try using Mono
http://www.go-mono.com/mono-downloads/download.html

Console Application

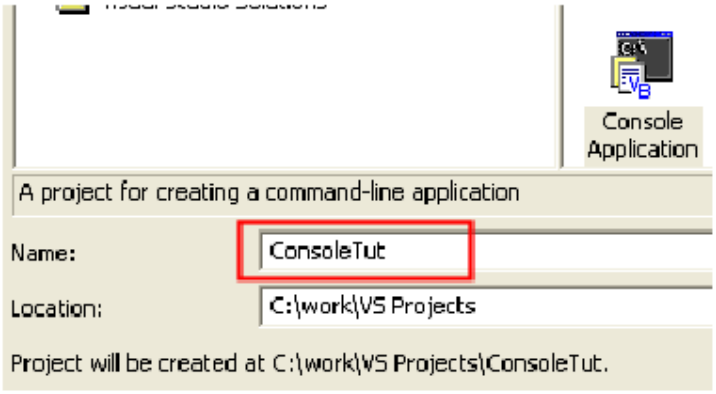
A VB.NET console application uses a command line window for its user interface. Because there is no opportunity for the user to provide inputs other than those asked for by the program, a console application is not "event driven". The actions performed in the program must follow in a linear fashion, and are completely defined by the program, and thus the programmer. This "procedural" programming is limiting, but makes a good introduction since the complexity of the user interface is eliminated.

1. Start Microsoft Visual Studio.NET. Note that it is not listed as Visual Basic.NET. Visual Studio.NET supports other programming languages other than VB. A Start Screen is displayed to help you open new or existing VB projects.

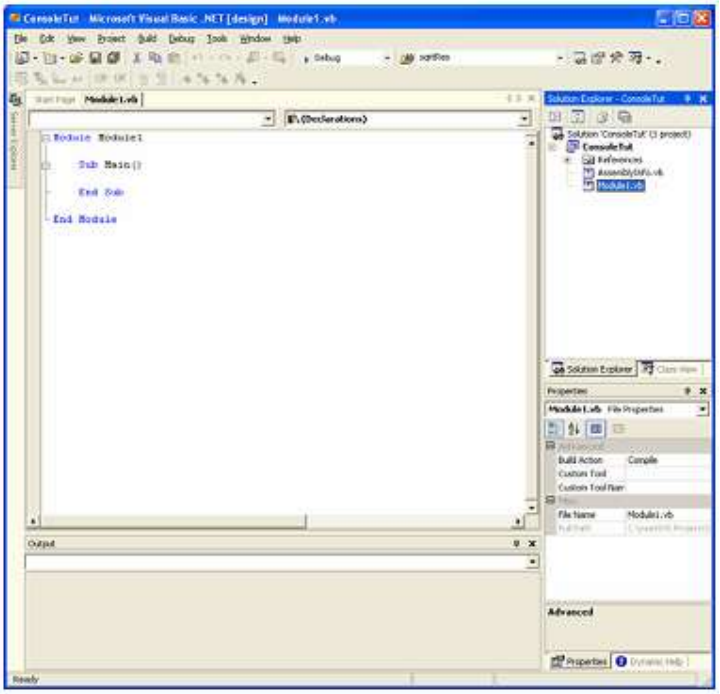
2. Click the New Project button.
3. The New Project dialog box is shown. You select the type of project and name it here.
4. Ensure that the Visual Basic projects folder is opened, and then scroll to and click Console Application from the list of templates.



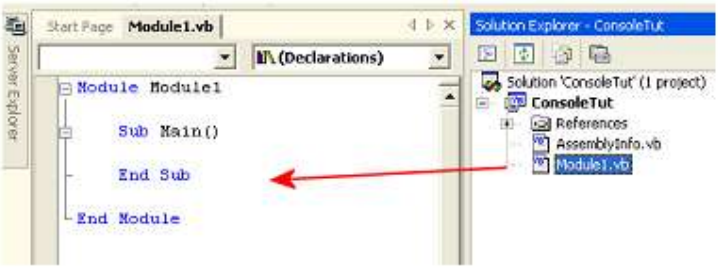
5. Enter ConsoleTut as the name of the project. All the files associated with the project will be stored in a folder as indicated in the Location text box (note this so you can find it later). Click OK to create the project.



6. The VS (Visual Studio) IDE (Integrated Development Environment) is where you build your program. You can actually create any VB program in Notepad, but the IDE provides a large number of tools that greatly simplifies the process of creating a Windows program.



7. For this tutorial, you can ignore most of the windows and tools in the IDE. The Solution Explorer in the top right corner of the IDE shows the files associated with the project. The console application template provides the required files. The Module1.vb file is the active file, and its contents are shown in the editor window.



8. The code contained within the Module1.vb file is listed between the two statements:

```
Module Module1

End Module
```

The module currently contains a single subroutine (don't worry too much about terms just yet) called Main. The code inside the Main subroutine is automatically "run" when the program starts. You will add all the code for this tutorial in between the Sub Main() and End Sub statements.

Tools: (Console.ReadLine for Input, Console.WriteLine for Output)

The console application simplifies input and output in that the user interface is essentially text based. The user types in the input, and the output is posted to the same text window. This text window is referenced in your program code by using the keyword Console. Three "actions" associated with the Console object enable you to receive input (ReadLine) and output (Write and WriteLine) text characters to the console window.

Basic data types

In order for a computer system to process and store data effectively, different kinds of data are formally given different types. This enables:

- data to be stored in an appropriate way, for example, as numbers or characters
- data to be manipulated effectively, for example numbers with mathematical operators and characters with concatenation
- automatic validation in some cases.

INTEGER

An **INTEGER** is a positive or negative whole number that can be used with mathematical operators.

SINGLE

A **Single** is a positive or negative number with a fractional part. Single numbers can be used with mathematical operators. (In pseudocode REAL data type is used for numbers with fractional part)

CHAR

A variable or constant of type **CHAR** is a single character.
Gender = 'F'

STRING

A variable or constant of type **STRING** is several characters in length. Strings vary in length and may even have no characters: an empty string. The characters can be letters and/or digits and/or any other printable symbol.

BOOLEAN

A **BOOLEAN** variable can have only two values: TRUE or FALSE.

Variables:

A variable is a named location in the computer's memory that stores a certain type of data (character, string of characters, integer, real number, and so on). The variable is identified in the program by a unique name.
The values stored in any variable are changed during execution of program.
By storing each of the inputs in memory (in a variable), you can use them again and again within the program without requiring that the user re-input them.

Variable Declaration:
Before variables can be used to store data, they must be “Declared”. These reserves the appropriate amount of memory to store the value assigned to the variable.
In the editor window, below Sub Main() but above End Sub, type:

```
DIM Name As String
DIM Weight1, Weight2, Weight_Difference  As Single
DIM Count As Integer
```

Constant:
A constant is a memory location in which stored values remain unchanged during execution of program.

Constant Declaration and Initialization:
Before constants can be used, they must be declared and initialized with permanent values. Declaration and assignment of permanent value is done in the same line like:

```
Const TaxRate as single=0.15
Const MaxMarks as single=100
```

Array:

Data stored in a computer is stored at any location in memory that the computer decides to use, which means that similar pieces of data can be scattered all over memory. This, in itself, doesn't matter to the user, except that to find each piece of data it has to be referred to by a name (known as a "variable name"). For example, if we want to store the 20 names of students in a group then each location would have to be given a different variable name. The first, "Abdullah", might be stored in location Name1, the second, "Rumaisa", might be stored inName2 , the third, "Rashid", could be stored in Name3.Creating the 20 different variable names is possible but these names are separate (as far as the computer is concerned) and do not relate to each other in any way. It would be far more sensible to force the computer to store them all together using the same variable name. However, this doesn't let me identify individual names. If I call the first one, Name(1) , the second one Name(2) and so on, it is obvious that they are all people's names and that they are distinguishable by their position in the list. A list like this is called an **array.**

Each element in the array is identified using its **subscript** or **index number**. The largest and smallest index numbers are called the *upper bound* and *lower bound* of the array.

	Name
1	Abdullah
2	Rumaisa
3	Rashid
4	Ahmed
5	Riaz
6	Patel
7	Smith
19	Mani
20	Ghayas

Declaring an array

It is important declare the arrays before assigning values in it so that program can reserve that amount of space in its memory; otherwise, there may not be enough space when the program uses the data.

Declaration consists of telling the computer program:

- the identifier name of the array
- the sort of data that is going to be stored in the array, i.e. its data type
- How many items of data are going to be stored, so that it knows how much space to reserve.

Different programming languages have different statements for initialising the array but they all do the same thing. In Visual Basic, the statement is:

Dim Name(20) As String

This Dim statement declares:

- the identifier name: Name
- the upper bound: 20
- the data type: String.

The upper bound of 20 specifies that there can be a maximum of 21 data items, since Visual Basic starts with a subscript of zero. We do not have to fill the array; the upper bound of 20 indicates the maximum size.

The array that has been described in one dimension array so far is really only a list of single data items. It is possible to have an array which can be visualised as a two-dimensional table with rows and columns and a data value in each cell.

Reading data into an array

To assign data values to the elements of the array, we do this with assignment statements such as:

Name(6) = “Patel”

This places the string “Allan” at index position 6 in the array.

Similarly, the following statement places the string “Rashid” at index position 3 in the array.

Name(19) = “Mani”

First program

Start a new Project and select Console Application. Type the following code:

```
module module1
    sub main()
        console. writeline("In the name of Allah")
        Console.writeline("Hello World!")
        console. readKey()
    end sub
end module
```

(Press F5 to run the code)

Tasks

0.1 – Write a program that displays the message "Asslam-o-Alaikum"

0.2 – Write a program that displays the message "My name is Muhammad and I live in Brussels" (replace Dave and Brussels with your own information)

0.3 – Write a program that displays the lyrics to your national anthem. Each line of the song should be on a fresh line.

Example Program 1 - Assignment - *Integer, byte, real, boolean, character, string, date/time.*

```
Module Module1
    Sub Main()
        Dim theNumber As Integer
        Dim theWord As String

        theWord = "Bird"
        theNumber = 9

        Console.WriteLine(theWord& " is the word")
        Console.WriteLine("And the number is " &theNumber)

        Console.ReadKey()

        theWord = "Cat"
        Console.WriteLine("Enter a number>")
        theNumber = Int(Console.ReadLine())
        Console.WriteLine("Now " &theWord& " is the word and the number is " &
        theNumber)

        Console.ReadKey()
    End Sub
End Module
```

Example Program 2 - Arithmetic - *+, –, /, x, DIV, MOD*

```
Module Module1
    Sub Main()
        Dim number1, number2, total As Integer

        Console.WriteLine("Enter first number")
        number1 = Int(Console.ReadLine())
        Console.WriteLine("Enter second number")
        number2 = Int(Console.ReadLine())
        total = number1 + number2
        Console.WriteLine("The total is " & total)

        Console.ReadKey()
    End Sub
End Module
```

Tasks

3.1) Write a program that divides a number entered by the user by 2

3.2) Write a program that displays the 7 times table

3.3) Write a program that displays any times table the user requests

Example Program 3 – Selection

```
Dim intInput As Integer

System. Console.WriteLine("Enter an integer…")
intInput = Val(System. Console.ReadLine())
If intInput = 1 Then
    System. Console.WriteLine("Thank you.")
ElseIfintInput = 2 Then
    System. Console.WriteLine("That's fine.")
ElseIfintInput = 3 Then
    System. Console.WriteLine("Too big.")
Else
    System. Console.WriteLine("Not a number I know.")
End If
Console.ReadKey()
```

Tasks

1. Write a program to ask the user what 24+9 is. Say “Excellent” if they get it right.
2. Write a program to ask the user “how many in a bakers dozen?” and say “most excellent” if they get it right.
3. Write a program to ask the user to enter their age. If their age is under 18 then say “Sorry, this film is not for you”.
4. Write a program to ask the user for two numbers. Compare the first with the second and then print out one of three messages. Either the numbers are equal, the first is bigger, or the second is bigger. You will need more than one IF to solve this one.

5. Write a program which asks the user to enter their password. If they enter the word "PASSWORD" then display the message "Welcome to the treasure", otherwise display a message which says "go away, it's all mine".
6. Write a program which asks the user to enter a number between 1 and 10. If the number entered is out with this range then display a message "Sorry…out of range".

Example Program 4 - Relational operators - =, <, >, <>, <=, >=

```
Module Module1
    Sub Main()
        Dim age As Integer
        Console.WriteLine("What is your age?")
        age = Int(Console.ReadLine())
        If age > 16 Then
            Console.WriteLine("You can drive!")
        Else
            Console.WriteLine("You are too young to drive")
        End If

        Console.ReadKey()
    End Sub
End Module
```

The symbols we can use to test for conditions are as follows:

<	Less than
<=	Less Than or Equal To
>	Greater than
>=	Greater Than or Equal To
==	IS Equal To
!= or <>	Not Equal To

Example Program 5 - Boolean operators - *NOT, AND, OR*

```
Module Module1
    Sub Main()
        Dim age, points As Integer
        Console.WriteLine("What is your age?")
        age = Int(Console.ReadLine())
        Console.WriteLine("How many points do you have on your licence?")
        points = Int(Console.ReadLine())
        If age > 16 And points < 9 Then
            Console.WriteLine("You can drive!")
        Else
            Console.WriteLine("You are not eligable for a driving licence")
        End If

        Console.ReadKey()
    End Sub
End Module
```

Example Program 6 - Built-in functions - *Arithmetic functions: round, truncation.*

```
Dim num As Double
Dim rounded As Integer
Dim squarert As Double
Dim trunc As Integer

Console.Write("Enter a real number")
num = Console.ReadLine()
rounded = Math.Round(num)
squarert = Math.Sqrt(num)
Console.WriteLine("round: " & rounded &vbNewLine& "Square Root: " &squarert)
trunc = Math.Truncate(num)
Console.WriteLine("The number truncated is " &trunc)
Console.WriteLine("This is not always the same as rounded")

Console.ReadKey()
```

Tasks

1) Write a program that asks for 5 numbers, calculates the mean average and then rounds it down. Display the result on screen.

Example Program 7 - String handling functions *length, position, substring, concatenation*.

```
Dim theString As String
theString = "Hello Dave, you're my wife now!"
Console.WriteLine(theString)
Console.WriteLine(theString.Length) 'display the string's length
Console.WriteLine(theString.ToUpper) 'display the string in upper case
Console.WriteLine(theString.ToLower) 'display the string in lower case
Console.WriteLine(theString.Contains("Dave")) 'is Dave there?
Console.WriteLine(theString.IndexOf("D")) 'position of D
Console.WriteLine(theString.Substring(12)) 'displays the substring starting at
position 12
Dim newString As String
newString = "Speak to Dave! " &theString 'string concatenation
Console.WriteLine(newString)
Console.ReadKey() ' pause and wait so user can read output. Tasks
```

Tasks

1. Write a program that checks a username against a stored value. How the user enters the username should NOT be case sensitive.
2. Adapt program 1 so that it also takes in a password. If the user enters spaces after the password the computer will trim them out automatically.
3. Write a program that will check a phone number is of the correct length.
4. Write a program that asks for a user's full name in one inputbox/textbox but then stores the first and second names in different variables.

Example Program 8 – Repetition

```
Module Module1
    Sub Main()
        Dim theNumber As Integer
        theNumber = 7
        'a loop
        For x = 1 To 10
            Console.WriteLine("7 x " & x & " = " & (7 * x))
        Next
        'the end of the loop
        Console.ReadKey() 'pause so user can see
    End Sub
End Module
```

Tasks

1. Write a program which asks for your name and then displays it 5 times on the screen.
2. Write a program to display the name of the town you live in 10 times.
3. Write a program to ask for a person's favourite CD and the artist. Both should be displayed on the same line 5 times.
4. Write a program to ask for a number and display its multiplication table 1 to 100
5. Write a program that asks the user for a number 5 times and adds them all up to give a total.

Example Program 9 - Constants

```
'a constant is a value that doesn't change
'using them greatly improves the readability of your code
'number constants
    Const conPi = 3.14159265358979

    Const conMaxPlanets As Integer = 9
    'string constants
    Const conVersion = "07.10.A"
    Const conCodeName = "Enigma"
    ' try assigning a new value to one of your constants
    'to make a constant available to the whole program "Public" should precede it
    'and it should be placed inside the module but outside the procedure
    'try it
    'there are string constants created by VB
    Console.WriteLine("Pi is" &conPi)
    Console.WriteLine("Pi is" &vbCr&conPi)
    Console.WriteLine("Pi is" &vbTab&conPi)
    Console.ReadKey()
```

Example Program 10 – 1 dimensional arrays

```
Dim countries(5) As String
Dim randomNum As Integer
countries(1) = "Scotland"
countries(2) = "Belgium"
countries(3) = "Netherlands"
countries(4) = "Germany"
countries(5) = "France"
Randomize()
randomNum = Int(Int((5 * Rnd()) + 1))
Console.WriteLine("You should go to " & countries(randomNum) & " on holiday.")
Console.ReadKey()
```

Tasks

1) Write a program which will set up an array to hold 50 numbers. Call the array numbers. Display
the array's contents across the screen. They should all be 0.
2) Create a program that stores an array of car records. At least 5 cars and 4 fields per record.
3) Create a program that stores an array of 5 people records. The information should be entered by
the user.
4) Adapt program 2 to now do a linear search for a certain car and display it's details.

Example Program 11 - Validation

```
Dim mark As Integer
Do
        Console.WriteLine("Enter a mark between 0 and 10")
        mark = Val(Console.ReadLine())
        If (mark > 10) Or (mark < 0) Then Console.WriteLine("That was not a valid mark")
Loop Until (mark >= 0) And (mark <= 10) ' keeps going until a valid mark is entered
Console.WriteLine("Well done!")
Console.ReadKey()
```

Tasks

1) Write a program that validates a user is old enough to drive (older than 17, younger than 80)
2) Write a program that checks that a telephone number entered is long enough (string length)
3) Write a program that checks that both a username and password is correct before allowing you to proceed.

Example Program 12 – Read from a text file

```
Dim objStreamReader As IO. StreamReader
Dim strLine As String
'Pass the file path and the file name to the StreamReader constructor.
objStreamReader = New IO. StreamReader("H: \Diary.txt")
'Read the first line of text.
strLine = objStreamReader.ReadLine
'Continue to read until you reach the end of the file.
Do While Not strLine Is Nothing
'Write the line to the Console window.
Console.WriteLine(strLine)
'Read the next line.
strLine = objStreamReader.ReadLine
Loop
'Close the file.
objStreamReader.Close()
Console.ReadLine()
```

Tasks

1) Write a program that reads the students' names from a txt file and displays them on the screen

2) Write a program that reads 10 team names from a txt file and stores them in an array

3) Write a program that reads 5 song titles from a csv file and displays them on the screen

4) Write a program that reads 20 team names from a csv file into an array, then displays the array on screen

Pre-release material May/June 2015

Here is a copy of the pre-release material

Task1

A school keep records of the weights of each pupil. The weight in kilograms of each pupil is recorded on the first day of term. Input and store the weights and names recorded for a class of 30 pupils. You must store the weights in a one-dimensional array and the names in another one-dimensional array. All the weights must be validated on entry and any invalid weights rejected. You must decide your own validation rules. You must assumethatthepupilsnamesareunique.Outputthenamesandweightsofthepupilsin the class.

Task2

The weight in kilograms of each pupil is recorded again onthelastdayofterm.Calculateandstorethedifferenceinweightforeachpupil.

Task3

Forthosepupilswhohaveadifferenceinweightofmorethan2.5kilograms,output, withasuitablemessage,thepupil'sname,thedifferenceinweightandwhetherthisis rise or a fall. Your program must include appropriate prompts for the entry of data.

Errormessagesandotheroutputsneedtobesetoutclearlyandunderstandably.All variables, constants and other identifiers must have meaningful names. Each task must be fully tested.

Coding for the given tasks

```
Module Module1

    Sub Main()
    Dim Name(30) As String
    Dim Count As Integer
    Dim Weight1(30) As Single

    Const Upper_Limit As Single = 500
    Const Lower_Limit As Single = 5


    'Task 1
    For Count = 1 To 30
        Console.WriteLine("Student No. : "& Count)
        Console.Write("Enter name  : ")
        Name(Count) = Console.ReadLine()
        Console.Write("Enter Weight at day 1 of term ")
        Weight1(Count) = Console.ReadLine()

    'Validation Check for Weight
    While Weight1(Count) < Lower_Limit Or Weight1(Count) > Upper_Limit
            Console.WriteLine("Error: Invalid weight. It must be between 5 and 500")
            Console.Write("Re-enter weight on first day ")
            Weight1(Count) = Console.ReadLine()
    EndWhile

    Next

    'For Displaying list of name and weight of students
    For Count = 1 To 5
        Console.WriteLine(Name(Count) &" "& Weight1(Count))
    Next

    'Task 2
    Dim weight2(30), Weight_Difference(30) AsSingle

    For Count = 1 To 30
        Console.WriteLine(Count &"   "& Name(Count) &"  "& Weight1(Count))
        Console.Write("Enter weight on last day  ")
        weight2(Count) = Console.ReadLine()
    'Validation Check for Weight
    While weight2(Count) < Lower_Limit Or weight2(Count) > Upper_Limit
            Console.WriteLine("Error: Invalid weight. It must be between 5 and 500")
            Console.Write("Re-enter weight on lastt day ")
            weight2(Count) = Console.ReadLine()
    EndWhile
        Weight_Difference(Count) = weight2(Count) - Weight1(Count)
```

```
        Next

        'task 3
        For Count = 1 To 30
        If Weight_Difference(Count) > 2.5 Then
                Console.WriteLine(Name(Count) &" has a rise in weight of "& Weight_Difference(Count)
        &" kg")
        ElseIf Weight_Difference(Count) < -2.5 Then
                Console.WriteLine(Name(Count) &" has a fall  in weight of "& Weight_Difference(Count)
        &" kg")

        EndIf
        Next


            Console.ReadKey()

        EndSub

EndModule
```

Sample Questions

Q1. (a)Declare the array to store each student's names and weights on day 1, weights last day and weight differences.

.....
.....
.....

(b) (i) Show the design of your algorithm to complete Task1using pseudo-code, programming statements or a flowchart.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

(ii)Comment on the efficiency of your design.

.....
.....
.....
.....
.....

(c) (i)Show the design of your algorithm to complete Task2 using pseudo-code, programming statements or a flowchart.

.....
.....
.....
.....
.....
.....
.....
.....
.....

.....
.....
.....
.....
.....

(ii) Comment on the efficiency of your design.

.....
.....
.....
.....
.....

(d) State validation rule you have used in task 1 and justify your rule.

Validation Rule: .....
.....

Justification: .....
.....

(e) Show two different sets of student data that you could use to check the validation used in Task 1. Explain why you chose each dataset.

Set1: .....
Reason for choice: .....
.....

Set 2: .....
Reason for choice: .....
.....

(f) (i) Explain how students with difference in weight will be chosen (Task 3). You may include pseudo-code or programming statements to help illustrate your explanation.

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

(ii)How does your program work if we need to know how many students have gained and lost their weights? You may include pseudo-code or programming statements to help illustrate your explanation.

.....
.....
.....
.....
.....
.....
.....

(g) A syntax error can occur when writing a program.

(i) State what is meant by a syntax error, giving an example.

.....
.....
.....
..... **[2]**

(ii) Describe tools and facilities available in an integrated development environment (IDE) which can help the programmer to identify and correct syntax errors.

.....
.....
.....
.....
.....
.....
.....
..... **[4]**