AQA Computer Science A-Level

4.1.1 Programming

Past Paper Mark Schemes

Additional Spec Qs AS Paper 1

03	**1**		**Marks are for AO2 (analysis)** The values are being stored as string; the string 007 is (alphabetically) less than 06;	2

03	**2**		**Mark is for AO3 (programming)**
			
			`IF Value1 < Value2 THEN`
			`  OUTPUT "Value 2 is larger"`
			`ELSE IF Value1 = Value2 THEN`
			`  OUTPUT "Value1 and Value2 are the same"`
			`ELSE`
			`  OUTPUT "Value 1 is larger"`
			`ENDIF`
			
			One mark - addition of check for equality and output message;
			One mark - statement works correctly;
				2

02	1	**Mark is for AO1 (knowledge)** A subroutine that calls itself;	1

02	2	**Mark is for AO2 (analyse)** When S is greater than or equal to E;	1

02	3	**All marks AO2 (apply)**	6

Call number	S	E	M	List returned
1	1	5	3	
2	1	3	2	
3	1	2	1	
4	1	1		[6]
3	1	2	1	
5	2	2		[3]
3	1	2	1	[6, 3]
2	1	3	2	
6	3	3		[4]
2	1	3	2	[6, 4, 3]
1	1	5	3	
7	4	5	4	
8	4	4		[8]
7	4	5	4	
9	5	5		[5]
7	4	5	4	[8, 5]
1	1	5	3	[8, 6, 5, 4, 3]

Mark as follows:
1 mark: Correct list returned by call number 3;
1 mark: Correct lists returned by call number 6-9;
1 mark: Correct list returned by call number 1;
1 mark: `S`, `M`, `E` given correct values for call number 6;
1 mark: `S`, `M`, `E` given correct values for call number 7;
1 mark: `S`, `M`, `E` given correct values for call number 8-9;

Info for examiner: Ignore missing values for `S`, `E`, `M` after the first time a particular call number appears in the table.

02	4	**Mark is for AO2 (analyse)** nlog n **A.** O(nlog n) **A.** $O(n\log_2 n)$ **A.** $n\log_2 n$ **NE.** log(n)	1

02	5	**All marks AO1 (knowledge)** Return address; Parameters; Local variables; Return value; **MAX 2**	2

02	6	**All marks AO2 (analyse)** Because there will be four recursive calls to `MergeSort` (1, 2, 3, 4) and (1, 2, 3, 5); there will also be three recursive calls to `MergeSort` (1, 2, 3) and one call to `Merge`.	2

June 2012 Comp 3

10	(d)	(i)	Routine defined in terms of itself // Routine that calls itself; **A** alternative names for routine e.g. procedure, algorithm **NE** repeats itself	1
10	(d)	(ii)	Stores return addresses; Stores parameters; Stores local variables; **NE** temporary variables Stores contents of registers; **A** To keep track of calls to subroutines/methods etc. **MAX 1** Procedures / invocations / calls must be returned to in reverse order (of being called); As it is a LIFO structure; **A** FILO As more than one / many return addresses / sets of values may need to be stored (at same time) // As the routine calls itself and for each call/invocation a new return address / new values must be stored; **MAX 1**	2

10	(e)						Discovered							Completely Explored							
				Call	V	U	EndV	1	2	3	4	5	6	7	1	2	3	4	5	6	7	F
					-	-	7	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
				DFS(1,7)	1	2	7	**T**	F	F	F	F	F	F	F	F	F	F	F	F	F	F
				DFS(2,7)	2	1	7	T	**T**	F	F	F	F	F	F	F	F	F	F	F	F	F
						3	7	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F
				DFS(3,7)	3	2	7	T	T	**T**	F	F	F	F	F	F	**T**	F	F	F	F	F
				DFS(2,7)	2	4	7	T	T	T	F	F	F	F	F	F	T	F	F	F	F	F
				DFS(4,7)	4	2	7	T	T	T	**T**	F	F	F	F	F	T	F	F	F	F	F
						5	7	T	T	T	T	F	F	F	F	F	T	F	F	F	F	F
				DFS(5,7)	5	4	7	T	T	T	T	**T**	F	F	F	F	T	F	F	F	F	F
						6	7	T	T	T	T	T	F	F	F	F	T	F	F	F	F	F
				DFS(6,7)	6	5	7	T	T	T	T	T	**T**	F	F	F	T	F	F	**T**	F	F
				DFS(5,7)	5	7	7	T	T	T	T	T	T	F	F	F	T	F	F	T	F	F
				DFS(7,7)	7	5	7	T	T	T	T	T	T	**T**	F	F	T	F	F	T	**T**	**T**
				DFS(5,7)	5	-	7	T	T	T	T	T	T	T	F	F	T	F	**T**	T	T	T
				DFS(4,7)	4	-	7	T	T	T	T	T	T	T	F	F	T	**T**	T	T	T	T
				DFS(2,7)	2	-	7	T	T	T	T	T	T	T	F	**T**	T	T	T	T	T	T
				DFS(1,7)	1	-	7	T	T	T	T	T	T	T	**T**	T	T	T	T	T	T	T
				1 mark for having the correct values changes in each region highlighted by a rectangle and no incorrect changes in the region. Ignore the contents of any cells that are not changed. **A** alternative indicators that clearly mean True and False. **A** it is not necessary to repeat values that are already set (shown lighter in table)																	5

June 2016 AS Paper 1

05	1	**All marks AO3 (programming)** 1. Correct variable declarations for `Value`, `Operation` and `Count`; **Note for examiners** If a language allows variables to be used without explicit declaration (eg Python) then this mark should be awarded if the three correct variables exist in the program code and the first value they are assigned is of the correct data type. 2. Correct prompt `"Enter integer (0-99): "`; 3. `Value` assigned value entered by user; 4. `WHILE` loop, with syntax allowed by the programming language and correct condition for the termination of the loop; 5. Correct syntax for the `IF` statement including condition and `ELSE` part; **A.** use of `ELSE IF` but condition must be correct 6. Correct syntax for `(Value DIV 10) + (Value MOD 10)` and/or `(Value DIV 10) * (Value MOD 10)`; 7. `Count` is given initial value `0` before iteration structure and incremented by `1` inside iteration structure; 8. Correct prompt `"The persistence is: "` and immediately followed by value of `Count` and after iteration structure; **Max 7** If code would not function correctly **Max 7** If brackets are missing for the multiplicative calculation at mark point 6. **I.** Case of variable names, strings and output messages **I.** Spacing in prompts **A.** Minor typos in variable names and output messages **A.** Initialisation of variables at declaration stage	8

05	2	**Mark is for AO3 (evaluate)**
		Info for examiners: must match code from 05.1, including prompts on screen capture matching those in code. Code for 05.1 must be sensible.
		
		First Test
		
		`Enter integer (0-99):`
		`47`
		`Calculate additive or multiplicative persistence (a or m)?`
		`m`
		`The persistence is:`
		`3`
		
		Second Test
		
		`Enter integer (0-99):`
		`77`
		`Calculate additive or multiplicative persistence (a or m)?`
		`a`
		`The persistence is:`
		`2`
		
		Mark as follows:
		Both tests showing provided test data being entered, correct choice being made (m/a) and correct persistence value being displayed ;
			1

05	3	**Mark is for AO2 (analysis)** The number of times the iteration/while loop will be performed is unknown (at the start of the loop) / not predetermined / indefinite; **N.E.** loop until a condition is met	1

VB.NET

05	1	Dim Value As Integer
		Dim Operation As Char
		Dim Count As Integer
		Console.Write("Enter integer (0-99): ")
		Value = Console.ReadLine()
		Console.Write("Calculate additive or multiplicative persistence (a or m)? ")
		Operation = Console.ReadLine()
		Count = 0
		While Value > 9
		  If Operation = "a" Then
		  Value = (Value \ 10) + (Value Mod 10)
		  Else
		  Value = (Value \ 10) * (Value Mod 10)
		  End If
		  Count = Count + 1
		End While
		Console.Write("The persistence is: ")
		Console.Write(Count)
		Console.ReadLine()
		
		NOTE: must be \ for integer division
			8

PASCAL

05	1		8

```
Var
  Value : Integer;
  Operation : Char;
  Count : Integer;
Begin
  Write('Enter integer (0-99): ');
  Readln(Value);
  Write('Calculate additive or multiplicative persistence (a or
m)?');
  Readln(Operation);
  Count := 0;
  While Value > 9 Do
  Begin
    If Operation = 'a' Then
      Value := (Value DIV 10) + (Value MOD 10)
    Else
      Value := (Value DIV 10) * (Value MOD 10);
    Count := Count + 1;
  End;
  Write('The persistence is: ')
  Write(Count);
  Readln();
End.
```

C#

05	**1**	`int Value, Count;`
		`string Operation;`
		`Console.WriteLine("Enter integer (0-99):");`
		`Value = Convert.ToInt32(Console.ReadLine());`
		`Console.WriteLine("Calculate the additive or multiplicative perisistence (a or m)?");`
		`Operation = Console.ReadLine();`
		`Count = 0;`
		`while (Value > 9)`
		`{`
		`  if (Operation == "a")`
		`  {`
		`  Value = (Value / 10) + (Value % 10);`
		`  }`
		`  else`
		`  {`
		`  Value = (Value / 10) * (Value % 10);`
		`  }`
		`  Count = Count + 1;`
		`}`
		`Console.Write("The persistence is: ");`
		`Console.WriteLine(Count);`
			8

JAVA

05	1	`AQAConsole2016 console = new AQAConsole2016();`
		
		`console.println("Enter integer (0-99): ");`
		`int value = console.readInteger("");`
		`console.println("Calculate additive or multiplicative persistence (a or m)? ");`
		`char operation =  console.readChar("");`
		
		`int count = 0;`
		
		`while (value > 9){`
		`  if(operation == 'a'){`
		`  value = (value/10) + (value%10);`
		`  }`
		`  else{`
		`  value = (value/10) * (value%10);`
		`  }`
		`  count += 1;`
		`}`
		
		`console.println("The persistence is: " + count);`
		
		A. Putting prompts inside `readInteger` and `readChar` rather than separate `println` statement
		A. Putting `count = count + 1` instead of `count += 1`.
		
		A. Answers that don't make use of the AQA classes
		
		NOTE: if a string is used for `operation` then the comparison would be `operation.equals("a")`
			8

PYTHON 2

05	1	`print "Enter integer (0-99): "`
		`Value = int(raw_input())`
		`print "Calculate additive or multiplicative persistence (a or m)? "`
		`Operation = raw_input()`
		`Count = 0`
		`while Value > 9:`
		`  if Operation == "a":`
		`  Value = (Value / 10) + (Value % 10)`
		`  else:`
		`  Value = (Value / 10) * (Value % 10)`
		`  Count = Count + 1`
		`print "The persistence is: "`
		`print Count`
		
		A. `Value = input()` rather than `Value = int(raw_input())`
		A. Putting prompts inside `raw_input` rather than separate `print` statement
			8

PYTHON 3

05	1	`print("Enter integer (0-99): ")`
		`Value = int(input())`
		`print("Calculate additive or multiplicative persistence (a or m)? ")`
		`Operation = input()`
		`Count = 0`
		`while Value > 9:`
		`  if Operation == "a":`
		`  Value = (Value // 10) + (Value % 10)`
		`  else:`
		`  Value = (Value // 10) * (Value % 10)`
		`  Count = Count + 1`
		`print("The persistence is: ")`
		`print (Count)`
		
		A. Putting prompts inside `raw_input` rather than separate `print` statement
		
		NOTE: Python 3 will require // to work for integer division
			8

June 2017 AS Paper 1

03	1	**All marks for AO3 (programming)** **Mark as follows:** 1. Correct prompts `"Enter a whole number: "` `"Enter another whole number: "` `Number1` and `Number2` assigned values entered by user; **R**. if inside loop 2. `Number1` and `Number2` assigned to `Temp1` and `Temp2` respectively; 3. `WHILE` loop with syntax allowed by the programming language and correct condition for termination of the loop; 4. Correct syntax and condition for the `IF` statement inside attempt at loop 5. Correct contents of `THEN` and `ELSE` part 6. Correct output `"… is GCF of … and …"` **A**. Temp1 instead of `Result` **A**. output on more than one line **R**. if inside loop **A**. variations on prompts **I**. minor differences in case and spelling **DPT**. If different identifiers	6

03	2	**Mark is for AO3 (evaluate)**
		
		****** SCREEN CAPTURE ******
		Must match code from 03.1, including prompts on screen capture matching those in code.
		Code for 03.1 must be sensible.
		
		Screen capture(s) showing the requested tests
		`>>>`
		`enter a whole number: 12`
		`enter another whole number: 39`
		`3 is GCF of 12 and 39`
		`>>>`
			1

03	3	**Mark is for AO2 (analyse)** to preserve the original values for later use // otherwise output won't make sense; *Note: must refer to the fact that original values are needed later*	1

Python 2

03	1	Number1 = int(raw_input("Enter a whole number: "))
		Number2 = int(raw_input("Enter another whole number: "))
		Temp1 = Number1
		Temp2 = Number2
		while Temp1 != Temp2:
		  if Temp1 > Temp2:
		  Temp1 = Temp1 - Temp2
		  else:
		  Temp2 = Temp2 - Temp1
		Result = Temp1
		print Result, " is GCF of ", Number1, " and ", Number2
			6

Python 3

03	1	Number1 = int(input("Enter a whole number: "))
		Number2 = int(input("Enter another whole number: "))
		Temp1 = Number1
		Temp2 = Number2
		while Temp1 != Temp2:
		  if Temp1 > Temp2:
		  Temp1 = Temp1 - Temp2
		  else:
		  Temp2 = Temp2 - Temp1
		Result = Temp1
		print(Result, " is GCF of ", Number1, " and ", Number2)
			6

VB.NET

03	1	Sub Main()
		  Dim Number1 As Integer
		  Dim Number2 As Integer
		  Dim Temp1 As Integer
		  Dim Temp2 As Integer
		  Dim Result As Integer
		  Console.Write("Enter a whole number: ")
		  Number1 = Console.ReadLine
		  Console.Write("Enter another whole number: ")
		  Number2 = Console.ReadLine
		  Temp1 = Number1
		  Temp2 = Number2
		  While Temp1 <> Temp2
		  If Temp1 > Temp2 Then
		  Temp1 = Temp1 - Temp2
		  Else
		  Temp2 = Temp2 - Temp1
		  End If
		  End While
		  Result = Temp1
		  Console.WriteLine(Result & " is GCF of " & Number1 & " and " & Number2)
		  Console.ReadLine()
		End Sub
			6

Pascal

03	1	program Project2;
		
		{$APPTYPE CONSOLE}
		
		uses
		  SysUtils;
		
		var
		  Number1, Number2 : Integer;
		  Temp1, Temp2 : Integer;
		  Result : Integer;
		begin
		  Write('Enter a whole number: ');
		  Readln(Number1);
		  Write('Enter another whole number: ');
		  Readln(Number2);
		  Temp1 := Number1;
		  Temp2 := Number2;
		  while Temp1 <> Temp2 do
		  if Temp1 > Temp2 then
		  Temp1 := Temp1 - Temp2
		  else
		  Temp2 := Temp2 - Temp1;
		  Result := Temp1;
		  Write(Result, ' is GCF of ', Number1, ' and ', Number2);
		  Readln;
		end.
			6

C#

03	1	static void Main(string[] args)
		{
		  int Number1 = 0, Number2 = 0;
		  int Temp1 = 0, Temp2 = 0;
		  int Result = 0;
		
		  Console.Write("Enter a whole number: ");
		  Number1 = Convert.ToInt32(Console.ReadLine());
		  Console.Write("Enter another whole number: ");
		  Number2 = Convert.ToInt32(Console.ReadLine());
		  Temp1 = Number1;
		  Temp2 = Number2;
		  while (Temp1 != Temp2)
		  {
		  if (Temp1 > Temp2)
		  {
		  Temp1 = Temp1 - Temp2;
		  }
		  else
		  {
		  Temp2 = Temp2 - Temp1;
		  }
		  }
		  Result = Temp1;
		  Console.WriteLine(Result + " is GCF of " + Number1 + " and
		" + Number2);
		  Console.ReadLine();
		}
			6

Java

03	**1**	`public static void main(String[] args)`
		`{`
		`  int Number1 = 0;`
		`  int Number2 = 0;`
		`  int Temp1 = 0;`
		`  int Temp2 = 0;`
		`  int Result = 0;`
		`  Number1 = Console.readInteger("Enter a whole number: ");`
		`  Number2 = Console.readInteger("Enter another whole number:`
		`");`
		`  Temp1 = Number1;`
		`  Temp2 = Number2;`
		`  while (Temp1 != Temp2)`
		`  {`
		`  if (Temp1 > Temp2)`
		`  {`
		`  Temp1 = Temp1 - Temp2;`
		`  }`
		`  else`
		`  {`
		`  Temp2 = Temp2 - Temp1;`
		`  }`
		`  }`
		`  Result = Temp1;`
		`  Console.println(Result + " is GCF of " + Number1 + " and "`
		`+ Number2 );`
		`}`
			6

June 2017 Paper 1

07	1	**4 marks for AO3 (design) and 8 marks for AO3 (programming)** **Mark scheme**	12

Level	Description	Mark Range
4	A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution that meets most of the requirements of **Task 1**. All of the appropriate design decisions have been taken. To award 12 marks, all of the requirements must be met.	10-12
3	There is evidence that a line of reasoning has been followed to produce a logically structured program. The program displays a prompt, inputs the string value and includes a loop. An attempt has been made to count the number of consecutive instances of a character and to output a character followed by the count of that character, although some of this may not work. The solution demonstrates good design work as most of the correct design decisions have been taken.	7-9
2	A program has been written and some appropriate, syntactically correct programming language statements have been written. There is evidence that a line of reasoning has been partially followed as although the program may not have the required functionality, it can be seen that the response contains some of the statements that would be needed in a working solution. There is evidence of some appropriate design work as the response recognises at least one appropriate technique that could be used by a working solution, regardless of whether this has been implemented correctly.	4-6
1	A program has been written and a few appropriate programming language statements have been written but there is no evidence that a line of reasoning has been followed to arrive at a working solution. The statements written may or may not be syntactically correct. It is unlikely that any of the key design elements of the task have been recognised.	1-3

Guidance

Evidence of AO3 (design) – 4 points:

Evidence of design to look for in responses:

1. Identifying that a method that looks at each character in text entered is needed
2. Identifying that a comparison is needed to check if the current character is the same as the previous character or not
3. Mechanism that "remembers" value of previous character in the string // mechanism that "remembers" character at start of the run
4. Identifying that the first character in the string can't be compared to a previous

		character // the last character in the string can't be compared to the next character **NOTE:** award mark based on method attempted in answer provided Note that AO3 (design) points are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works. **Evidence for AO3 (programming) – 8 points:** Evidence of programming to look for in response: 5. Suitable prompt displayed before any loop structures 6. Text input by user and stored into a variable with a suitable name, after prompt is displayed and before any loop structures 7. Loop structure coded with correct termination condition 8. Selection structure coded with correct condition, selection structure must be inside loop **A.** second loop structure with correct condition that is nested in first loop structure 9. One added to count of character under the correct circumstances 10. Count of character reset to one under the correct circumstances 11. Character and correct count of character displayed for some characters from beginning of text input by user 12. Character and correct count of character displayed for all characters of any text entered by the user Note that AO3 (programming) points are for programming and so should only be awarded for syntactically correct code. **Information for examiner:** Refer answers that use alternative methods to produce the RLE to team leader.	
07	**2**	**Mark is for AO3 (evaluate)** *****SCREEN CAPTURE(S)***** *Info for examiner: Must match code from 7.1, including prompts on screen capture matching those in code. Code for 7.1 must be sensible.* Display of suitable prompt and user input of `AAARRRRGGGHH` followed by output of `A 3 R 4 G 3 H 2;` **A.** Each output on its own line, no spaces, other delimiter used instead of space	**1**
07	**3**	**Mark is for AO3 (evaluate)** *****SCREEN CAPTURE(S)***** *Info for examiner: Must match code from 7.1, including prompts on screen capture matching those in code. Code for 7.1 must be sensible.* Display of suitable prompt and user input of `A` followed by output of `A 1;` **A.** no space between `A` and `1`, other delimiter used instead of space	**1**

VB.NET

07	1	**Example Solution**
		
		`Sub Main()`
		`  Dim Text As String`
		`  Dim LastChar As String`
		`  Dim CountOfLastChar As Integer`
		`  Console.Write("Enter the text to compress: ")`
		`  Text = Console.ReadLine()`
		`  Console.Write("The compressed text is: ")`
		`  LastChar = ""`
		`  CountOfLastChar = 0`
		`  For Count = 0 To Len(Text) - 1`
		`  If Text(Count) = LastChar Then`
		`  CountOfLastChar += 1`
		`  Else`
		`  If LastChar <> "" Then`
		`  Console.Write(LastChar & " " & CountOfLastChar & " ")`
		`  End If`
		`  LastChar = Text(Count)`
		`  CountOfLastChar = 1`
		`  End If`
		`  Next`
		`  Console.Write(LastChar & " " & CountOfLastChar & " ")`
		`  Console.ReadLine()`
		`End Sub`
			12

Python 2

07	1	`text = raw_input("Enter the text to compress: ")`
		`print "The compressed text is:",`
		`LastChar = ""`
		`CountOfLastChar = 0`
		`for Count in range(0, len(text)):`
		`  if text[Count] == LastChar:`
		`  CountOfLastChar += 1`
		`  else:`
		`  if LastChar != "":`
		`  print LastChar,  CountOfLastChar,`
		`  LastChar = text[Count]`
		`  CountOfLastChar = 1`
		`print LastChar,CountOfLastChar`
			12

Python 3

07	1	**Example Solution**
		
		`text = input("Enter the text to compress: ")`
		`print ("The compressed text is: ", end="")`
		`LastChar = ""`
		`CountOfLastChar = 0`
		`for Count in range(0, len(text)):`
		`    if text[Count] == LastChar:`
		`        CountOfLastChar += 1`
		`    else:`
		`        if LastChar != "":`
		`            print (LastChar,  " " , CountOfLastChar, " ",end="")`
		`        LastChar = text[Count]`
		`        CountOfLastChar = 1`
		`print (LastChar,  " " , CountOfLastChar, " ")`
			12

C#

07	1	`string Text = "";`
		`string LastChar = "";`
		`int CountOfLastChar = 0;`
		`Console.Write("Enter the text to compress: ");`
		`Text = Console.ReadLine();`
		`Console.Write("The compressed text is: ");`
		`for (int Count = 0; Count < Text.Length ; Count++)`
		`{`
		`  if (Text[Count].ToString() == LastChar )`
		`  {`
		`  CountOfLastChar++;`
		`  }`
		`  else`
		`  {`
		`  if (LastChar != "")`
		`  {`
		`  Console.Write(LastChar + " " + CountOfLastChar + " ");`
		`  }`
		`  LastChar = Text[Count].ToString();`
		`  CountOfLastChar = 1;`
		`  }`
		`}`
		`Console.Write(LastChar + " " + CountOfLastChar + " ");`
		`Console.ReadLine();`
			12

Pascal

07	1	**Example solution**	12

```
var
  Text : string;
  LastChar : string;
  CountOfLastChar : integer;
  Count : integer;
begin
  write('Enter the text to compress: ');
  readln(Text);
  write('The compressed text is: ');
  LastChar := '';
  CountOfLastChar := 0;
  for Count := 1 to Length(Text) do
    begin
      if Text[Count] = LastChar then
        inc(CountOfLastChar)
      else
        begin
          if LastChar <> '' then
            write(LastChar, ' ', CountOfLastChar, ' ');
          LastChar := Text[Count];
          CountOfLastChar := 1;
        end;
    end;
  write(LastChar, ' ', CountOfLastChar, ' ');
  readln;
end.
```

Java

07	**1**	`public static void main(String[] args)`
		`{`
		`String Text;`
		`char LastChar;`
		`int CountOfLastChar;`
		`Console.print("Enter the text to compress: ");`
		`Text = Console.readLine();`
		`Console.print("The compressed text is: ");`
		`LastChar = ' ';`
		`CountOfLastChar = 0;`
		`for (int Count = 0; Count < Text.length(); Count++)`
		`{`
		`char CurrentChar = Text.charAt(Count);`
		`if(CurrentChar == LastChar)`
		`{`
		`CountOfLastChar += 1;`
		`}`
		`else`
		`{`
		`if (LastChar !=' ')`
		`{`
		`Console.print(LastChar + " " + CountOfLastChar + "`
		`");`
		`}`
		`LastChar = CurrentChar;`
		`CountOfLastChar = 1;`
		`}`
		`}`
		`Console.print(LastChar + " " + CountOfLastChar + " ");`
		`Console.readLine();`
		`}`
			12

June 2011 Comp 1

7	20	**VB.Net**
		`Sub Main()`
		`  Dim Names(4) As String`
		`  Dim Current As Integer`
		`  Dim Max As Integer`
		`  Dim Found As Boolean`
		`  Dim PlayerName As String`
		
		`  Names(1) = "Ben"`
		`  Names(2) = "Thor"`
		`  Names(3) = "Zoe"`
		`  Names(4) = "Kate"`
		`  Max = 4`
		`  Current = 1`
		`  Found = False`
		`  Console.WriteLine("What player are you looking for?")`
		`  PlayerName = Console.ReadLine`
		`  While Found = False And Current <= Max`
		`  If Names(Current) = PlayerName Then`
		`  Found = True`
		`  Else`
		`  Current = Current + 1`
		`  End If`
		`  End While`
		`  If Found = True Then`
		`  Console.WriteLine("Yes, they have a top score")`
		`  Else`
		`  Console.WriteLine("No, they do not have a top score")`
		`  End If`
		`  Console.ReadLine()`
		`End Sub`	

		Mark as follows: Correct variable declarations for `Max`, `Current`, `Found`, `PlayerName` and correct declaration for the `Names` array; Four correct values assigned to the correct positions in the `Names` array; `Max`, `Current`, `Found` initialised correctly; Correct prompt followed by `PlayerName` assigned value entered by user; WHILE loop formed correctly and correct conditions for the termination of the loop; First IF followed by correct condition and IF statement is inside the loop; THEN followed by correct assignment statement within a correctly formed IF statement; ELSE followed by correct assignment statement within a correctly formed IF statement; Second IF followed by correct condition and IF is after the loop; THEN followed by correct output within a correctly formed IF statement; ELSE followed by correct output within a correctly formed IF statement; **I.** Case of variable names, player names and output messages	

Pascal

```
Program Question7;
Var
  Names : Array[1..4] Of String;
  Current : Integer;
  Max : Integer;
  Found : Boolean;
  PlayerName : String;
Begin
  Names[1] := 'Ben';
  Names[2] := 'Thor';
  Names[3] := 'Zoe';
  Names[4] := 'Kate';
  Max := 4;
  Current := 1;
  Found := False;
  Writeln('What player are you looking for?');
  Readln(PlayerName);
  While (Found = False) And (Current <= Max)
    Do
      Begin
        If Names[Current] = PlayerName
          Then Found := True
          Else Current := Current + 1;
      End;
  If Found = True
    Then Writeln('Yes, they have a top score')
    Else Writeln('No, they do not have a top
score');
  Readln;
End.
```

C# Mark Scheme

7	20	namespace Question7
		{
		class Program
		{
		public static string[] Names = new string[5];
		public static int Current;
		public static int Max;
		public static bool Found;
		public static string PlayerName;
		
		static void Main(string[] args)
		{
		Names[1] = "Ben";
		Names[2] = "Thor";
		Names[3] = "Zoe";
		Names[4] = "Kate";
		Max = 4;
		Current = 1;
		Found = false;
		Console.WriteLine("What player are you looking for?");
		PlayerName = Console.ReadLine();
		while (!Found && Current <= Max)
		{
		if (Names[Current] == PlayerName)
		Found = true;
		else
		Current++;
		};
		if (Found)
		Console.WriteLine("Yes, they have a top score");
		else
		Console.WriteLine("No, they do not have a top score");
		Console.WriteLine("Press the Enter key to continue";
		Console.ReadLine();
		}
		}
		}
		
		A. Declaring and initialising a variable in one statement
		A. Variable declarations without `Public` keyword
			11

Java Mark Scheme

7	20	public class Question7 {
		
		  AQAConsole console = new AQAConsole();
		
		  public Question7() {
		  String[] names = new String[5];
		  int max;
		  int current;
		  boolean found;
		  String playerName;
		
		  names[1] = "Ben";
		  names[2] = "Thor";
		  names[3] = "Zoe";
		  names[4] = "Kate";
		
		//possible alternative, which declares and
		//instantiates in one.
		//String[] names={"","Ben","Thor","Zoe","Kate"};
		
		  current = 1;
		  max = 4;
		  found = false;
		
		  playerName = console.readLine("What player are
		you looking for? ");
		  while ((found == false) && (current <= max)) {
		  if (names[current].equals(playerName)) {
		  found = true;
		  } else {
		  current++;
		  } // end if/else
		  } // end while
		
		  if (found == true) {
		  console.println("Yes, they have a top
		score");
		  } else {
		  console.println("No, they do not have a top
		score");
		  } // end if/else
		  }// end CONSTRUCTOR
		
		  /**
		  * @param args the command line arguments
		  */
		  public static void main(String[] args) {
		  new Question7();
		  }
		}
			11

		A. Minor typos in variable names and output messages **A.** `Max` declared as a constant instead of a variable **A.** Alternative conditions with equivalent logic for the loop **A.** Array positions 0-3 used instead of 1-4 if consistent usage throughout program	**11**
	21	****SCREEN CAPTURE**** *Must match code from 20, including prompts on screen capture matching those in code. Code for 20 must be sensible.* **Mark as follows:** 'What player are you looking for?' + user input of 'Thor'; 'Yes, they have a top score' message shown; **I.** spacing **R.** If code for 20 would not produce this test run	**2**
	22	****SCREEN CAPTURE**** *Must match code from 20, including prompts on screen capture matching those in code. Code for 20 must be sensible.* **Mark as follows:** 'What player are you looking for?' + user input of 'Imran'; 'No, they do not have a top score' message shown; **I.** spacing **R.** If code for 20 would not produce this test run	**2**

June 2012 Comp 1

	17	****SCREEN CAPTURE**** *Must match code from 16, including prompts on screen capture matching those in code* **Mark as follows:** 'Enter bit value: ' + first user input of 1 'Enter bit value: ' + second user input of 1; 'Enter bit value: ' + third user input of 0 'Enter bit value: ' + fourth user input of 1; Value of 13 outputted;	**3**
	18	15;	**1**

PASCAL Mark Scheme

Qu	Part	Marking Guidance	Marks

6	16	`Program Question6;`
		`  Var`
		`  Answer : Integer;`
		`  Column : Integer;`
		`  Bit : Integer;`
		`  Begin`
		`  Answer := 0;`
		`  Column := 8;`
		`  Repeat`
		`  Writeln('Enter bit value: ');`
		`  Readln(Bit);`
		`  Answer := Answer + (Column * Bit);`
		`  Column := Column DIV 2;`
		`  Until Column < 1;`
		`  Writeln('Decimal value is: ', Answer);`
		`  Readln;`
		`  End.`
			11

VB.NET Mark Scheme

Qu	Part	Marking Guidance	Marks

6	16	`Sub Main()`
		`  Dim Answer As Integer`
		`  Dim Column As Integer`
		`  Dim Bit As Integer`
		`  Answer = 0`
		`  Column = 8`
		`  Do`
		`  Console.Write("Enter bit value: ")`
		`  Bit = Console.ReadLine`
		`  Answer = Answer + (Column * Bit)`
		`  Column = Column / 2`
		`  Loop Until Column < 1`
		`  Console.Write("Decimal value is: " & Answer)`
		`  Console.ReadLine()`
		`End Sub`
		
		Alternative Answer
		`Column = Column \ 2`
			11

JAVA Mark Scheme

Qu	Part	Marking Guidance	Marks

6	16	`public class Question6 {`
		`  AQAConsole console=new AQAConsole();`
		`  public Question6(){`
		`  int column;`
		`  int answer;`
		`  int bit;`
		`  answer=0;`
		`  column=8;`
		`  do{`
		`  console.print("Enter bit value: ");`
		`  bit=console.readInteger("");`
		`  answer=answer+(column*bit);`
		`  column=column/2;`
		`  }while(column>=1);`
		`  console.print("Decimal value is: ");`
		`  console.println(answer);`
		`  }`
		`  public static void main(String[] arrays){`
		`  new Question6();`
		`  }`
		`}`
			11

PYTHON Mark Scheme

Qu	Part	Marking Guidance	Marks

6	16	**# Section B Q6 Python 2.6**
		`Answer = 0`
		`Bit = 0`
		`Column = 8`
		
		`while Column >= 1:`
		`  print "Enter bit value: "`
		`  # Accept: Bit = int(raw_input("Enter bit value: "))`
		`  Bit = input()`
		`  Answer = Answer + (Column * Bit)`
		`  Column = Column // 2`
		`print "Decimal value is: ", Answer`
		`# or + str(Answer)`
		
		# Section B Q6 Python 3.1
		`Answer = 0`
		`Bit = 0`
		`Column = 8`
		
		`while Column >= 1:`
		`  print("Enter bit value: ")`
		`  # Accept: Bit = int(input("Enter bit value: "))`
		`  Bit = int(input())`
		`  Answer = Answer + (Column * Bit)`
		`  Column = Column // 2`
		`print("Decimal value is: " + str(Answer))`
		`# or print("Decimal value is: {0}".format(Answer))`
		
		A. `Answer` and `Bit` not declared at start as long as they are spelt correctly and when they are given an initial value that value is of the correct data type
			11

June 2013 Comp 1

19	Correct variable declarations for `Guess`, `NumberOfGuesses` and `NumberToGuess`; Correct prompt "`Player One enter your chosen number:` "; Followed by `NumberToGuess` assigned value entered by the user; 1st loop has syntax allowed by the programming language and one correct condition for the termination of the loop; **A.** alternative correct logic for condition 1st loop has syntax allowed by the programming language and has 2nd correct condition for the termination of the loop; **A.** alternative correct logic for condition Correct prompt "`Not a valid choice, please enter another number:` " followed by `NumberToGuess` assigned value entered by the user – must be inside the 1st iteration structure; `Guess` and `NumberOfGuesses` initialised correctly; 2nd loop has syntax allowed by the programming language and both correct conditions for the termination of the loop and is after the initialising of `Guess` and `NumberOfGuesses`; **A.** alternative correct logic for conditions Correct prompt "`Player Two have a guess:` " followed by `Guess` assigned value entered by the user – must be inside the 2nd iteration structure; `NumberOfGuesses` incremented inside the 2nd iteration structure; `If` statement with correct condition – must not be in an iterative structure; Correct output message in `Then` part of selection structure; Correct output message in `Else` part of selection structure; **I.** Case of variable names and output messages **A.** Minor typos in variable names and output messages **I.** spacing in prompts **A.** initialisation of variables at (or immediately after) declaration stage	**13**

20	****SCREEN CAPTURE**** *Must match code from 19, including prompts on screen capture matching those in code. Code for 19 must be sensible.* **Mark as follows:** 'Player One enter your chosen number: ' + user input of `0` 'Not a valid choice, please enter another number: ' Message shown;	**4**
	user input of `11` 'Not a valid choice, please enter another number: ' Message shown; user input of `5` 'Player Two have a guess: ' + user input of `5`; 'Player Two wins' message shown; **R.** If no evidence of user input **A.** alternative output messages if match code for 19	

21	****SCREEN CAPTURE**** *Must match code from 19, including prompts on screen capture matching those in code. Code for 19 must be sensible.* **Mark as follows:** 'Player One enter your chosen number: ' + user input of `6`; 'Player Two have a guess: ' + user input of `1` 'Player Two have a guess: ' + user input of `3` 'Player Two have a guess: ' + user input of `5` 'Player Two have a guess: ' + user input of `7` 'Player Two have a guess: ' + user input of `10`; 'Player One wins' message shown; **R.** If no evidence of user input **A.** alternative output messages if match code for 19	**3**
22	If a `FOR` loop was used then Player Two will always have 5 guesses // a `WHILE` loop will mean that the loop will terminate when Player Two guesses correctly // the number of times to iterate is not known before the loop starts;	**1**

Specimen AS Paper 1

01	1	**All marks AO3 (programming)**
		
		Python 2.6:
		
		`print "How far to count?"`
		`HowFar = input()`
		`while HowFar < 1:`
		`  print "Not a valid number, please try again."`
		`  HowFar = input()`
		`for MyLoop in range(1,HowFar+1):`
		`  if MyLoop%3 == 0 and MyLoop%5 == 0:`
		`  print "FizzBuzz"`
		`  elif MyLoop%3 == 0:`
		`  print "Fizz"`
		`  elif MyLoop%5 == 0:`
		`  print "Buzz"`
		`  else:`
		`  print MyLoop`
		
		1 mark: Correct prompt "`How far to count?`" followed by `HowFar` assigned value entered by user;
		1 mark: `WHILE` loop has syntax allowed by the programming language and correct condition for the termination of the loop;
		1 mark: Correct prompt "`Not a valid number, please try again.`" followed by `HowFar` being assigned value entered by the user (must be inside iteration structure);
		1 mark: Correct syntax for the `FOR` loop using correct range appropriate to language;
		1 mark: Correct syntax for an `IF` statement, including a `THEN` and `ELSE/ELIF` part;
		1 mark: Correct syntax for `MyLoop MOD 5 = 0` and `MyLoop MOD 3 = 0` used in the `IF` statement(s);
		1 mark: Correct output for cases in the selection structure where `MyLoop MOD 3 = 0` or `MyLoop MOD 5 = 0` or both - outputs "`FizzBuzz`", "`Fizz`" or "`Buzz`" correctly;
		1 mark: Correct output (in the `ELSE` part of selection structure), when `MyLoop MOD 3 <> 0` and `MyLoop MOD 5 <> 0` - outputs value of `MyLoop`;
			8

01	2	**All marks AO3 (evaluate)** **Info for examiners:** must match code from 01.1, including prompts on screen capture matching those in code. Code for 01.1 must be sensible.	1

		First Test
		
		`How far to count?`
		`18`
		`1`
		`2`
		`Fizz`
		`4`
		`Buzz`
		`Fizz`
		`7`
		`8`
		`Fizz`
		`Buzz`
		`11`
		`Fizz`
		`13`
		`14`
		`FizzBuzz`
		`16`
		`17`
		`Fizz`
		
		Second Test
		
		`How far to count?`
		`-1`
		`Not a valid number, please try again.`
		
		Screenshot with user input of `18` and correct output and user input of -1 and correct output;
		
		A. different formatting of output eg line breaks	

01	**3**	**Mark is for AO2 (analysis)** A FOR loop is used as it is to be repeated a known number of times;	1

01	**4**	**All marks AO2 (analysis)** Example of input: [nothing input] [a string] for example: `12A` Method to prevent: can protect against by using a try,except structure // exception handling; test the input to see if digits only;	3

		convert string to integer and capture any exception; use a repeat/while structure // alter repeat/while to ask again until valid data input; **1 mark:** Example of input **Max 2 marks:** Description of how this can be protected against	

C#

01	**1**	`int HowFar;`
		`int MyLoop;`
		`Console.WriteLine("How far to count?");`
		`HowFar = int.Parse(Console.ReadLine());`
		`while (HowFar < 1)`
		`{`
		`  Console.WriteLine("Not a valid number, please try again.");`
		`  HowFar = int.Parse(Console.ReadLine());`
		`}`
		`for (MyLoop = 1; MyLoop < HowFar+1; MyLoop++)`
		`{`
		`  if (MyLoop % 3 == 0 && MyLoop % 5 == 0)`
		`  {`
		`  Console.WriteLine("FizzBuzz");`
		`  }`
		`  else`
		`  {`
		`  if (MyLoop % 3 == 0)`
		`  {`
		`  Console.WriteLine("Fizz");`
		`  }`
		`  else`
		`  {`
		`  if (MyLoop % 5 == 0)`
		`  {`
		`  Console.WriteLine("Buzz");`
		`  }`
		`  else`
		`  Console.WriteLine(MyLoop);`
		`}`
			8

Java

01	1	`int howFar;`
		`int myLoop;`
		`Scanner in = new Scanner(System.in);`
		`System.out.println("How far to count?");`
		`System.out.println("Enter i Value:  ");`
		`howFar = in.nextInt();`
		`while (howFar < 1)`
		`{`
		`  System.out.println("Not a valid number, please try again.");`
		`  howFar = in.nextInt();`
		`}`
		`for (myLoop = 1; myLoop < howFar+1; myLoop++)`
		`{`
		`  if (myLoop % 3 == 0 && myLoop % 5 == 0)`
		`  {`
		`  System.out.println("FizzBuzz");`
		`  }`
		`  else`
		`  {`
		`  if (myLoop % 3 == 0)`
		`  {`
		`  System.out.println("Fizz");`
		`  }`
		`  else`
		`  {`
		`  if (myLoop % 5 == 0)`
		`  {`
		`  System.out.println("Buzz");`
		`  }`
		`  else`
		`  System.out.println(myLoop);`
		`  }`
		` }
			8

If students use the AQAConsole module, a possible solution :

```
int howFar;
int myLoop;
console.println("How far to count?");
howFar = console.readInteger();
while (howFar < 1)
{
    console.println("Not a valid number, please try again.");
    howFar = console.readInteger());
```

```
}
for (myLoop = 1; myLoop < howFar+1; myLoop++)
{
    if (myLoop % 3 == 0 && myLoop % 5 == 0)
    {
        console.println("FizzBuzz");
    }
    else
    {
        if (myLoop % 3 == 0)
        {
            console.println("Fizz");
        }
        else
        {
            if (myLoop % 5 == 0)
            {
                console.println("Buzz");
            }
            else
                console.println(myLoop);
       }
    }
}
```

Pascal

01	**1**	`program FizzBuzz(input,output);`
		
		`var`
		`  HowFar,MyLoop : Integer;`
		
		`begin`
		`  writeln('How far to count?');`
		`  readln(HowFar);`
		`  while HowFar < 1 Do`
		`  readln(HowFar);`
		`  for MyLoop := 1 to HowFar do`
		`  begin`
		`  if (MyLoop Mod 3 = 0) And (MyLoop Mod 5 = 0)`
		`then`
		`  writeln('FizzBuzz')`
		`  else if MyLoop Mod 3 = 0 then`
		`  writeln('Fizz')`
		`  else if MyLoop Mod 5 = 0 then`
		`  writeln('Buzz')`
		`  else writeln(MyLoop);`
		`  end;`
		
		`end.`
			8

VB. Net

01	1	Module Module1
		
		Sub Main()
		Dim HowFar As Integer
		Dim MyLoop As Integer
		Console.WriteLine("How far to count?")
		HowFar = Console.ReadLine()
		While HowFar < 1
		Console.WriteLine("Not a valid number, please try again.")
		HowFar = Console.ReadLine()
		End While
		For MyLoop = 1 To HowFar
		If MyLoop Mod 3 = 0 And MyLoop Mod 5 = 0 Then
		Console.WriteLine("FizzBuzz")
		ElseIf MyLoop Mod 3 = 0 Then
		Console.WriteLine("Fizz")
		ElseIf MyLoop Mod 5 = 0 Then
		Console.WriteLine("Buzz")
		Else
		Console.WriteLine(MyLoop)
		End If
		Next
		Console.ReadLine()
		End Sub
		
		End Module
			8

Python 3

01	1	print ("How far to count?")
		HowFar = int(input())
		while HowFar < 1:
		print ("Not a valid number, please try again.")
		HowFar = int(input())
		for MyLoop in range(1,HowFar+1):
		if MyLoop%3 == 0 and MyLoop%5 == 0:
		print ("FizzBuzz")
		elif MyLoop%3 == 0:
		print ("Fizz")
		elif MyLoop%5 == 0:
		print ("Buzz")
		else:
		print (MyLoop)
			8

Specimen Paper 1

06	1	**4 marks for AO3 (design) and 8 marks for AO3 (programming)** **Mark Scheme**	12

Level	Description	Mark Range
4	A line of reasoning has been followed to arrive at a logically structured working or almost fully working programmed solution that meets all of the requirements of **Task 1** and some of the requirements of **Task 2**. All of the appropriate design decisions have been taken. To award 12 marks, all of the requirements of both tasks must be met.	10-12
3	There is evidence that a line of reasoning has been followed to produce a logically structured program. The program displays a prompt, inputs the decimal value and includes a loop, which might be a definite or indefinite loop. An attempt has been made to do the integer division, output the remainder within the loop and use the result of the division for the next iteration, although some of this may not work. The solution demonstrates good design work as most of the correct design decisions have been taken. To award 9 marks, all of the requirements of **Task 1** must have been met.	7-9
2	A program has been written and some appropriate, syntactically correct programming language statements have been written. There is evidence that a line of reasoning has been partially followed as although the program may not have the required functionality for either task, it can be seen that the response contains some of the statements that would be needed in a working solution to **Task 1**. There is evidence of some appropriate design work as the response recognises at least one appropriate technique that could be used by a working solution, regardless of whether this has been implemented correctly.	4-6
1	A program has been written and a few appropriate programming language statements have been written but there is no evidence that a line of reasoning has been followed to arrive at a working solution. The statements written may or may not be syntactically correct. It is unlikely that any of the key design elements of the task have been recognised.	1-3

Guidance

Task 1:

Evidence of AO3 (design) - 3 points:

Evidence of design to look for in responses:

- Identifying that an indefinite loop must be used (as the length of the input is variable)
- Identifying the correct Boolean condition to terminate the loop
- Correct identification of which commands belong inside and outside the loop

Note that AO3 (design) points are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works.

Evidence of AO3 (programming) – 6 points:

Evidence of programming to look for in responses:

- Prompt displayed
- Value input by user and stored into a variable with a suitable name
- Loop structure coded
- Remainder of integer division calculated
- Remainder of integer division output to screen
- Result of integer division calculated and assigned to variable so that it will be used in the division operation for the next iteration

Note that AO3 (programming) points are for programming and so should only be awarded for syntactically correct code.

Task 2:

Evidence of AO3 (design) - 1 point:

Evidence of design to look for in responses:

- A sensible method adopted for reversing the output eg appending to a string or storing into an array

Note that AO3 (design) points are for selecting appropriate techniques to use to solve the problem, so should be credited whether the syntax of programming language statements is correct or not and regardless of whether the solution works.

Evidence of AO3 (programming) – 2 points:

Evidence of programming to look for in responses:

- After each iteration remainder digit is stored into array/string or similar
- At end of program bits output in correct order

Note that AO3 (programming) points are for programming and so should only be awarded for syntactically correct code.

Example Solution VB.Net

Task 1:

```
Dim DecimalNumber As Integer
Dim ResultOfDivision As Integer
Dim BinaryDigit As Integer

Console.WriteLine("Please enter decimal number to
convert")
DecimalNumber = Console.ReadLine

Do
    ResultOfDivision = DecimalNumber \ 2
    BinaryDigit = DecimalNumber Mod 2
    Console.Write(BinaryDigit)
    DecimalNumber = ResultOfDivision
Loop Until ResultOfDivision = 0
```

Task 2:

```
Dim DecimalNumber As Integer
Dim ResultOfDivision As Integer
Dim BinaryDigit As Integer
Dim BinaryString As String

Console.WriteLine("Please enter decimal number to
convert")
DecimalNumber = Console.ReadLine
BinaryString = ""
```

```
Do
     ResultOfDivision = DecimalNumber \ 2
     BinaryDigit = DecimalNumber Mod 2
     BinaryString = BinaryDigit.ToString() +
BinaryString
     DecimalNumber = ResultOfDivision
Loop Until ResultOfDivision = 0

Console.WriteLine(BinaryString)
```

Example Solution Pascal

Task 1:

```
Var
  DecimalNumber, ResultOfDivision, BinaryDigit :
Integer;

Begin
  Writeln('Please enter decimal number to convert');
  Readln(DecimalNumber);
  Repeat
     ResultofDivision := DecimalNumber Div 2;
     BinaryDigit := DecimalNumber Mod 2;
     Write(BinaryDigit);
     DecimalNumber := ResultOfDivision;
  Until ResultOfDivison = 0;
  Readln;
End.
```

Task 2:

```
Var
  DecimalNumber, ResultOfDivision, BinaryDigit :
Integer;
  BinaryString : String;

Begin
  Writeln('Please enter decimal number to convert');
  Readln(DecimalNumber);
  BinaryString := '';
  Repeat
     ResultofDivision := DecimalNumber Div 2;
     BinaryDigit := DecimalNumber Mod 2;
     BinaryString := IntToStr(BinaryDigit) +
BinaryString;
     DecimalNumber := ResultOfDivision;
  Until ResultOfDivision = 0;
  Writeln(BinaryString);
  Readln;
```

Example Solution Python 3.x

Task 1:

```
print("Input a decimal number to convert to binary:
", end = '')
decimal = int(input())

while decimal != 0:
    print(decimal % 2, end = '')
    decimal //= 2
```

Task 2:

```
print("Input a decimal number to convert to binary:
", end = '')
decimal = int(input())

result = ""
while decimal != 0:
    result = str(decimal % 2) + result
    decimal //= 2

print(result)
```

Alternative answers using break:

Task 1:

```
print("Input a decimal number to convert to binary:
", end = '')
decimal = int(input())

while True:
    print(decimal % 2, end = '')
    decimal //= 2
    if decimal == 0:
        break
```

Task 2:

```
print("Input a decimal number to convert to binary:
", end = '')
decimal = int(input())

result = ""
while True:
    result = str(decimal % 2) + result
    decimal //= 2

    if decimal == 0:
        break

print(result)
```

Example Solution Python 2.x

Task 1:

```
print "Input a decimal number to convert to
binary:",
decimal = int(input())

while decimal != 0:
    print decimal % 2,
    decimal /= 2
```

Task 2:

```
print "Input a decimal number to convert to
binary:",
decimal = int(input())

result = ""
while decimal != 0:
    result = str(decimal % 2) + result
    decimal /= 2

print(result)
```

Alternative answers using break:

Task 1:

```
print "Input a decimal number to convert to
binary:",
decimal = int(input())

while True:
    print decimal % 2,
    decimal /= 2
    if decimal == 0:
        break
```

Task 2:

```
print "Input a decimal number to convert to
binary:",
decimal = int(input())

result = ""
while True:
    result = str(decimal % 2) + result
    decimal /= 2

    if decimal == 0:
        break

print result
```

Example Solution C#

Task 1:

```
int DecimalNumber;
int ResultOfDivision;
int BinaryDigit;

Console.WriteLine("Please enter decimal number to
convert");
DecimalNumber = int.Parse(Console.ReadLine());
do
{
   ResultOfDivision = DecimalNumber / 2;
   BinaryDigit = DecimalNumber % 2;
   Console.Write(BinaryDigit);
   DecimalNumber = ResultOfDivision;
} while (ResultOfDivision != 0);
```

Task 2:

```
int DecimalNumber;
int ResultOfDivision;
int BinaryDigit;
string BinaryString;

Console.WriteLine("Please enter decimal number to
convert");
DecimalNumber = int.Parse(Console.ReadLine());
BinaryString = "";
do
{
   ResultOfDivision = DecimalNumber / 2;
   BinaryDigit = DecimalNumber % 2;
   BinaryString = Convert.ToString(BinaryDigit) +
BinaryString;
   DecimalNumber = ResultOfDivision;
} while (ResultOfDivision != 0);
Console.WriteLine(BinaryString);
```

Example Solution Java

Task 1:

```
int decimalNumber;
int resultOfDivision;
int binaryDigit;
```

```
decimalNumber = console.readInteger("Please enter
decimal number to convert");
do {
   resultOfDivision = decimalNumber / 2;
   binaryDigit = decimalNumber % 2;
   console.print(binaryDigit);
   decimalNumber = resultOfDivision;
} while (resultOfDivision != 0);
```

Task 2:

```
int decimalNumber;
int resultOfDivision;
int binaryDigit;
String binaryString;

decimalNumber = console.readInteger("Please enter
decimal number to convert");
binaryString = "";
do {
   resultOfDivision = decimalNumber / 2;
   binaryDigit = decimalNumber % 2;
   binaryString = Integer.toString(binaryDigit) +
binaryString;
   decimalNumber = resultOfDivision;
} while (resultOfDivision != 0);
console.println(binaryString);
```