Student responses with examiner commentary

V1.0 29/10/2014

Student responses with examiner commentary

A-level Computer Science 7517
Paper 1 7517/1E

For teaching from September 2015

For assessment from summer 2017

Specimen assessment paper 1 7517/1E

Introduction

These resources should be used in conjunction with the Specimen Assessment material (7517/E) from the AQA website. This document illustrates how examiners intend to apply the mark scheme in live papers. While every attempt has been made to show a range of student responses examiners have used responses, and subsequent comments, which will provide teachers with the best opportunity to understand the application of the mark scheme. Examples given in this commentary use VB.Net

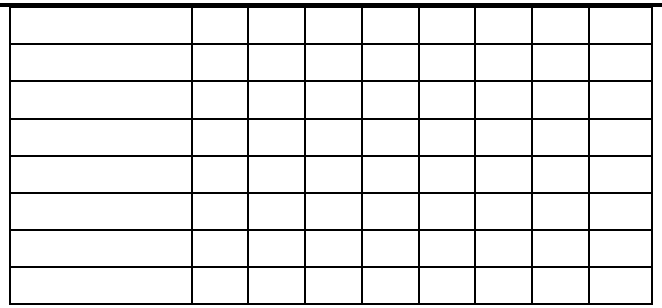
Specimen Paper 1 – Example of marked exam paper: Student 2

<table>
<tr><th>Question Number</th><th>Student answer</th><th>Marks awarded</th><th>Marks available</th><th>Commentary</th></tr>
<tr><td>01.1</td><td>A or B</td><td>0</td><td>1</td><td>Mark can’t be awarded, even though it does contain the correct answer, as more than one answer given.</td></tr>
<tr><td>01.2</td><td>Nathan was not killed by poison so it was not Ian and Nathan was not killed by a blow on the neck so it was not Suzanne. So it was either Steve or Paul.</td><td>1</td><td>2</td><td>“Nathan was not killed by poison” is worth a mark. This is the only creditworthy point so 2nd mark not awarded.</td></tr>
<tr><td>02.1</td><td><table><tr><th>Original state</th><th>Input</th><th>New state</th></tr><tr><td>S3</td><td>0</td><td>S4</td></tr><tr><td>S3</td><td>1</td><td>S2</td></tr></table></td><td>1</td><td>1</td><td>Correct answer.</td></tr>
<tr><td>02.2</td><td>(0|1)*(00|11)* (0|1)*</td><td>2</td><td>3</td><td>There is (0|1)* at the start of the regular expression and (0|1)* at the end of the regular expression – so that is 2 marks. The (00|11) is correct but the * is wrong as this part is not optional – there must be at least one instance of either 00 or 11 for the string to be accepted.</td></tr>
<tr><td>02.3</td><td><table><tr><th>Rule number (given in Figure 2)</th><th>Could be defined using a regular expression</th></tr><tr><td>1</td><td>Y</td></tr><tr><td>2</td><td>Y</td></tr><tr><td>3</td><td>Y</td></tr><tr><td>4</td><td>N</td></tr><tr><td>5</td><td>N</td></tr><tr><td>6</td><td>Y</td></tr></table></td><td>1</td><td>1</td><td>Correct answer.</td></tr>
</table>

<table>
<tr><td>02.4</td><td>It is missing part of the rule, it should say
<word> ::= <char><word>|<char></td><td>1</td><td>2</td><td>The rule has been modified correctly but the explanation does not make it clear what was wrong with the original rule.</td></tr>
<tr><td>03.1</td><td>Because it is not connected.</td><td>0</td><td>1</td><td>While a graph cannot be a tree if it is not connected, this graph is connected so this is not the reason why this particular graph is not a tree.</td></tr>
<tr><td>03.2</td><td><table><tr><th>Vertex (in Figure 3)</th><th>Adjacent vertices</th></tr><tr><td>1</td><td>2, 3</td></tr><tr><td>2</td><td>1, 3</td></tr><tr><td>3</td><td>1, 2, 5</td></tr><tr><td>4</td><td>2</td></tr><tr><td>5</td><td>3</td></tr></table></td><td>1</td><td>2</td><td>All rows are correct except for vertex 2 where ‘4’ is missing from the adjacent vertices – so 1 mark is awarded.</td></tr>
<tr><td>03.3</td><td>It is better to use an adjacency list when there are not many lines between the vertices.</td><td>1</td><td>2</td><td>The use of the word ‘line’ instead of ‘edge’ is fine. However, only one reason has been given.</td></tr>
<tr><td>03.4</td><td><table><tr><th></th><th></th><th></th><th></th><th colspan="5">Cat</th></tr><tr><th>NoOfCats</th><th>A</th><th>B</th><th>C</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th></tr><tr><td>5</td><td>2</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td></td><td>3</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td>4</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td>5</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table></td><td>3</td><td>6</td><td>A is set in the correct sequence.

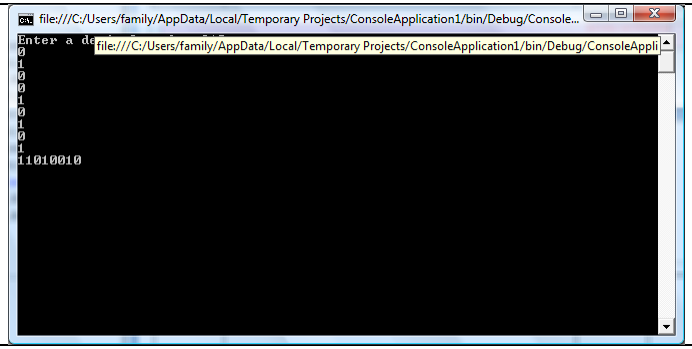
The marks for the correct values of B and C are not awarded.

The student has worked out the correct values for NoOfCats, Cat[1], Cat[4] and Cat[5] – but not the correct values for Cat[2] and Cat[3].</td></tr>
</table>

				
03.5	To put the cats into order.	0	1	This is not the correct answer.
03.6	A problem that can be solved but only when the problem is big it takes too long.	2	2	Both marks awarded.
03.7	Use heuristics to come up with a solution that is good but not the best.	2	2	The phrasing of the answer isn't perfect but both marks have been awarded. To be completely accurate it should say "not **guaranteed** to be the best".
04.1	False	1	1	Correct answer.
04.2	Then	0	1	Not enough for the mark to be given.
04.3	L ← M + 1	1	2	Not the correct answer. L being assigned a value is worth 1 mark.
04.4	O(kn)	1	1	The n should be superscripted – but the meaning is clear enough for the mark to be given.
04.5	O(log n)	1	1	Correct answer.
04.6	O(1)	1	1	Correct answer.
04.7	O(n)	1	1	Correct answer.
04.8	Because if there are n items in the list then you might not find the one you are looking for until you have looked at every item in the list as it could be the last item you look at. This means it could take n comparisons to get the item wanted.	2	2	Answer is worth 2 marks. The student has assumed that the item being looked for is in the list when it is also possible that the item is not in the list - however, this does not invalidate the explanation given.

05.1	3 * 4	1	1	Correct answer.
05.2	(12 + 8) * 4	1	1	Correct answer.
05.3	Can be used by computers	0	1	While true this answer is not sufficient for the mark to be awarded.

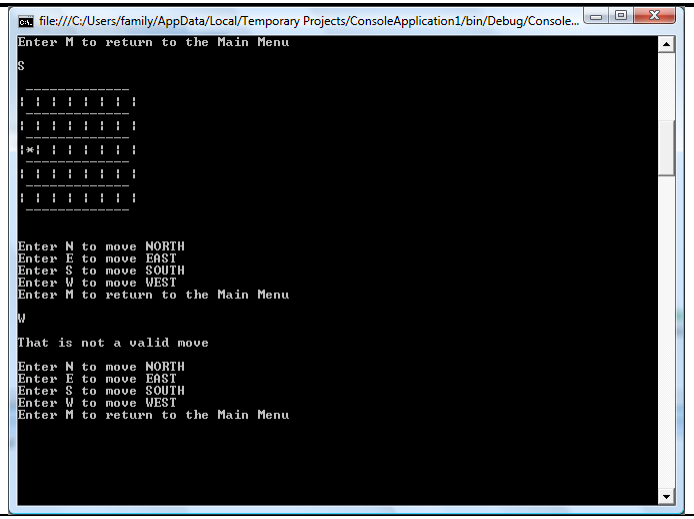
06.1	`Sub Main()`
	`  Dim DecimalNumber As Integer`
	`  Dim Remainder As Integer`
	`  Console.Write("Enter a decimal number ")`
	`  DecimalNumber = Console.ReadLine`
	`  While DecimalNumber > 0`
	`  Remainder = DecimalNumber Mod 2`
	`  DecimalNumber = DecimalNumber / 2`
	`  Console.WriteLine(Remainder)`
	`  End While`
	`  Console.WriteLine(11010010)`
	`  Console.ReadLine()`
	`End Sub`
		8	12	Answer written using VB.Net programming language.
				
				Task 1 – A03 (design)
				The answer shows good evidence of design (it covers all three points). Student has identified that an indefinite loop is needed, has identified the correct condition for the termination of the loop and also identified which statements belong inside the loop. The design is appropriate for Task 1 but makes it difficult to complete Task 2 correctly as it does not store the earlier remainders so that they can be reversed at a later stage.
				
				Task 1 – AO3 (programming)
				Of the six evidence points outlined in the mark scheme, 5 are implemented correctly in the student's answer. They have not used integer division to calculate the new value of `DecimalNumber` (`/` should be used instead of `\`).
				
				Task 2 – AO3 (design)
				No evidence provided.
				
				Task 2 – AO3 (programming)
				The program does output the correct bit pattern at the end – but this has not been achieved in a sensible way (it will always output the bit pattern for the denary number 210 no matter what value was entered by the user).
				No evidence of any program code that stores the remainder

				in a string or array.
06.2		1	2	Suitable prompt displayed and correct test data entered so first mark is awarded. Program does not calculate the correct bit pattern (due to an error in the method used for the division). The correct reversed bit pattern (for 210) is displayed but code that achieves this is not sensible.
07.1	The arrows are pointing in the wrong direction. There is no Monster class.	2	2	Both answers correct – does not state what the class should be instead of `Monster` but this is not necessary for the mark to be awarded.
07.2	`Trap1`	0	1	Has identified an object that is in the Skeleton program but this is not an example of instantiation (the whole line of code `Dim Trap1 As New Trap` is needed as without this it is not clear that the student has understood what instantiation is).
07.3	`CavernState`	1	1	Correct answer.
07.4	`Trap`	1	1	Correct answer.
07.5	`Count`	1	1	Correct answer.
07.6	`Enemy`	0	1	The only class that uses composition in the Skeleton Program is `Game`.
07.7	To make sure that the flag or the monster can't be in the same square as the player.	1	1	Meaning is clear – though it would be more accurate to add "at the start of a new game".
07.8	Named constants make it easier to update programs	1	2	One of the possible correct answers. No second answer given so only 1 mark can be awarded.
07.9	You would need to add the line `Dim Trap3 As New`	1	2	While the answer does not specify where the line should be

	Trap. You would also need to adjust the game so that it uses the new trap.			added the mark can still be given. The second half of the answer is too vague to be creditworthy – details of how to adjust the game need to be given for the 2nd mark to be awarded.

08.1	`Public Function CheckValidMove(ByVal Direction As Char) As Boolean`
	`  Dim ValidMove As Boolean`
	`  ValidMove = True`
	`  If Not (Direction = "N" Or Direction = "S" Or Direction = "W" Or Direction = "E" Or Direction = "M") Then`
	`  ValidMove = False`
	`  End If`
	`  If Direction = "W" And Player.GetPosition.NoOfCellsEast <= 0 Then`
	`  ValidMove = False`
	`  End If`
	`  Return ValidMove`
	`End Function`
		3	3	Answer written using VB.Net programming language.
				
				Condition is not exactly the same as the one in the mark scheme – but will give the correct functionality. Fully-working answer.

08.2	`Do`
	`  DisplayMoveOptions()`
	`  MoveDirection = GetMove()`
	`  ValidMove = CheckValidMove(MoveDirection)`
	`  If ValidMove = False Then`
	`  Console.WriteLine("That is not a valid move")`
	`  End If`
	`Loop Until ValidMove`
		1	2	Answer written using VB.Net programming language.
				
				IF statement has been added in the correct place in the code and has a condition that is equivalent to the one shown in the Mark Scheme. However, the error message shown is not the one asked for in the question.

<table>
<tr>
<td>08.3</td>
<td> 
</td>
<td>1</td>
<td>1</td>
<td>The screen capture shows that the player is at the western end of the cavern, has attempted to then move to the west and the error message has been displayed. The error message is not the one asked for in the question – but this does not matter in 8.3 as long as it does match the error message shown in the answer for 8.2 (which it does).</td>
</tr>
<tr>
<td>09.1</td>
<td><pre>
Class SleepyEnemy
  Inherits Enemy
  Protected MovesTillSleep As Integer

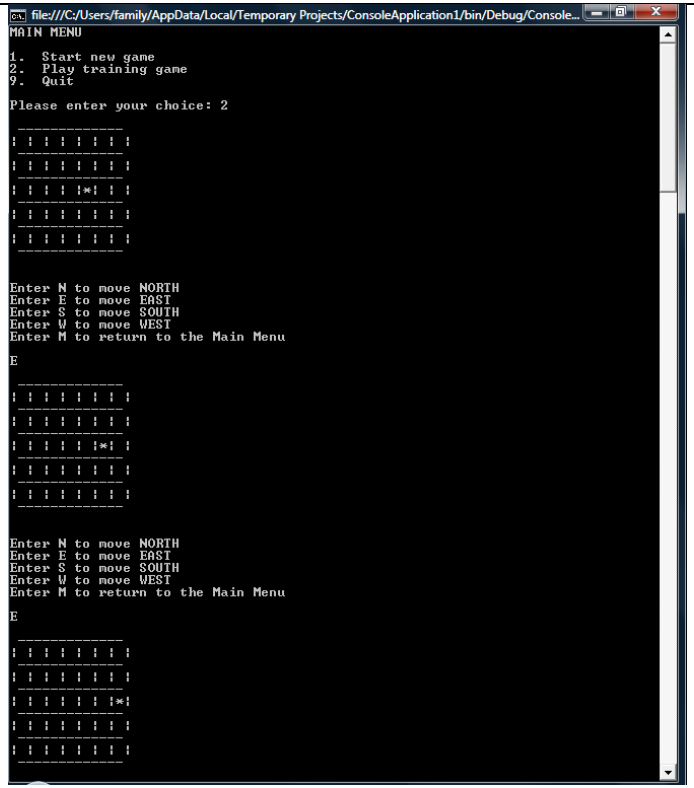
  Public Overrides Sub ChangeSleepStatus()
    MovesTillSleep = 4
  End Sub

  Public Overrides Sub MakeMove(ByVal
PlayerPosition As CellReference)
    MyBase.MakeMove(PlayerPosition)
    MovesTillSleep -= 1
    If MovesTillSleep = 0 Then
      ChangeSleepStatus()
    End If
  End Sub
End Class
</pre></td>
<td>7</td>
<td>8</td>
<td>Answer written using VB.Net programming language.

All changes have been made correctly – except for the overriding of <code>ChangeSleepStatus</code>. This method has been overridden but the existing functionality of the method in the parent class has not been kept as the line <code>MyBase.ChangeSleepStatus()</code> is missing – this means that, in this example, the overriding has not been done correctly (in the way specified in the question).

The student will know (from their testing) that the code is not fully correct but has done the sensible thing and included the code that they have written even though it is not fully correct.

<code>MovesTillSleep</code> has been set as protected instead of private – this is allowed as it provides the same functionality</td>
</tr>
</table>

				within this program.
09.2		0	2	The player has correctly moved East twice and there is a message saying that the monster is now awake. However, there is no evidence of the monster moving two squares nearer to the player so the 1st mark has not been awarded. The player has then moved one square to the south. There is no evidence of the monster moving a further two squares nearer to the player. At the end the monster would appear to be asleep but there is no evidence that it was awake so the 2nd mark has not been awarded. The student's program fails because the overriding `ChangeSleepStatus` does not call the overridden `ChangeSleepStatus`.

10.1	`Public Sub DisplayMoveOptions()`
	`  Console.WriteLine()`
	`  Console.WriteLine("Enter N to move NORTH")`
	`  Console.WriteLine("Enter E to move EAST")`
	`  Console.WriteLine("Enter S to move SOUTH")`
	`  Console.WriteLine("Enter W to move WEST")`
	`  Console.WriteLine("Enter M to return to`
		1	1	Answer written using VB.Net programming language.
				
				The message does not match that shown in the mark scheme but that does not matter in this question as the exact message to use was not specified in the question (so any appropriate message is fine).

	the Main Menu")
	  Console.WriteLine("Enter A to use an arrow")
	  Console.WriteLine()
	End Sub			

10.2	Public Function CheckValidMove(ByVal Direction As Char) As Boolean
	  Dim ValidMove As Boolean
	  ValidMove = True
	  If Not (Direction = "N" Or Direction = "S" Or Direction = "W" Or Direction = "E" Or Direction = "M" Or Direction = "A") Then
	  ValidMove = False
	  End If
	  If Direction = "W" And Player.GetPosition.NoOfCellsEast = 0 Or Direction = "A" And Player.GetHasArrow = False Then
	  ValidMove = False
	  End If
	  Return ValidMove
	End Function
		2	2	Answer written using VB.Net programming language.
				
				The student has added option A to the first IF statement. They have then added correct extra conditions to their answer to an earlier question to make it only valid to shoot an arrow if the player has an arrow (instead of doing this as a separate IF statement).
				
				Code has all the required functionality so gets both marks.

10.3	Class Character
	  Inherits Item
	  Private HasArrow As Boolean
	
	  Public Sub MakeMove(ByVal Direction As Char)
	  Select Case Direction
	  Case "N"
	  NoOfCellsSouth = NoOfCellsSouth - 1
	  Case "S"
	  NoOfCellsSouth = NoOfCellsSouth +
		6	8	Answer written using VB.Net programming language.
				
				The program code is mostly correct but there are two errors in it. Firstly, the arrow direction check is missing a condition in the loop (N appears twice, W does not appear at all) – this means that a direction of W will not be accepted when shooting the arrow. Secondly, the assignment statement that changes the `HasArrow` property to `False` is after the `Return` statement meaning that the player will be able to shoot unlimited arrows.

```
1
      Case "W"
        NoOfCellsEast = NoOfCellsEast - 1
      Case "E"
        NoOfCellsEast = NoOfCellsEast + 1
    End Select
  End Sub

  Public Sub New()
    HasArrow = True
  End Sub

  Public Function GetHasArrow()
    Return HasArrow
  End Function

  Function GetArrowDirection()
    Console.WriteLine()
    Console.WriteLine("Enter N to shoot to
the NORTH")
    Console.WriteLine("Enter E to shoot to
the EAST")
    Console.WriteLine("Enter S to shoot to
the SOUTH")
    Console.WriteLine("Enter W to shoot to
the WEST")
    Dim Choice As Char
    Do
      Choice = Console.ReadLine
      If Not (Choice = "N" Or Choice = "E"
Or Choice = "S" Or Choice = "W") Then
        Console.WriteLine("Not a valid
direction enter a different direction")
      End If
    Loop Until Choice = "N" Or Choice =
"E" Or Choice = "S" Or Choice = "N"
    Return Choice
```

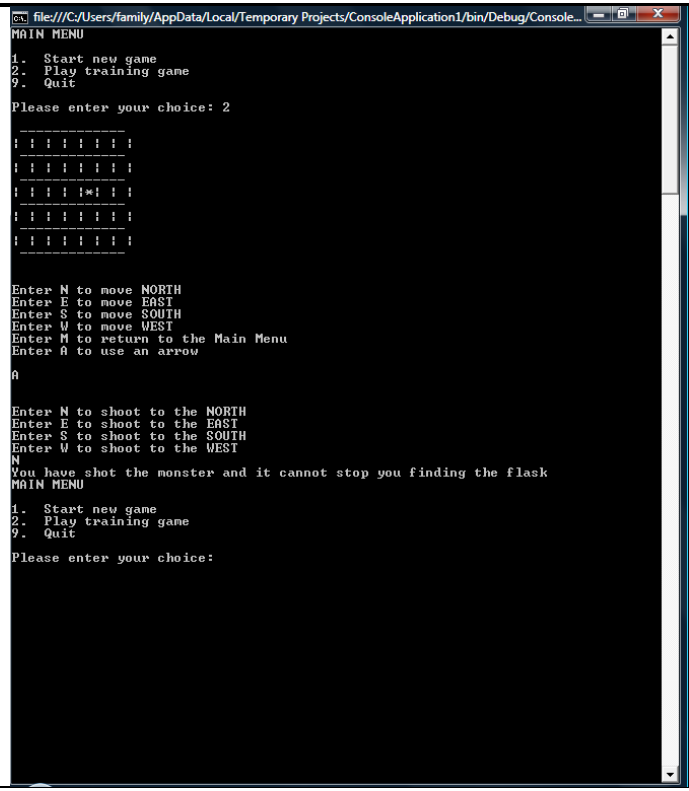
No return type is given for `GetHasArrow` and `GetArrowDirection` but, assuming Option Strict is off in VB.Net this relaxed typing is allowed and the missing return data type will not prevent correct functionality.

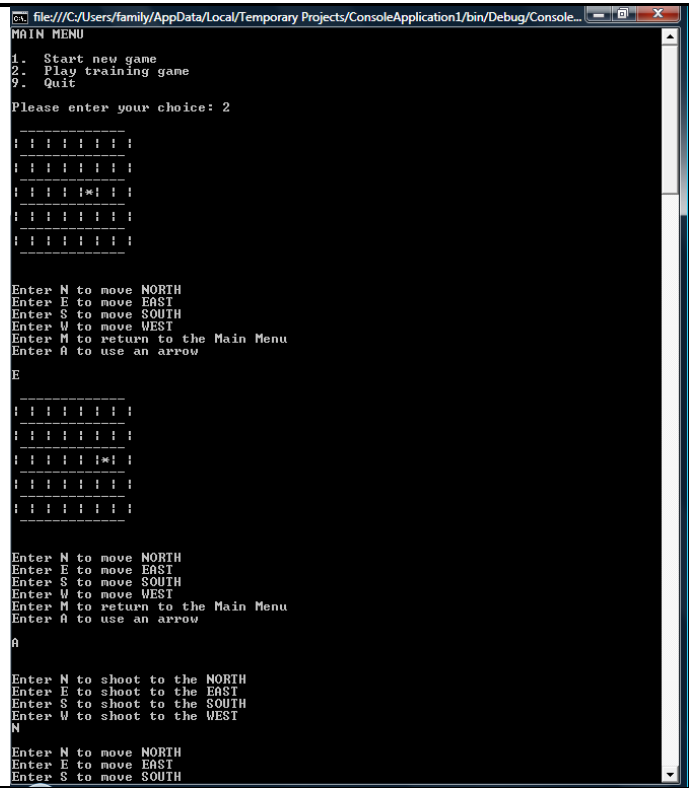
	HasArrow = False End Function End Class			

10.4	Public Sub Play()
	Dim Count As Integer
	Dim Eaten As Boolean
	Dim FlaskFound As Boolean
	Dim MoveDirection As Char
	Dim ValidMove As Boolean
	Dim Position As CellReference
	Eaten = False
	FlaskFound = False
	Cavern.Display(Monster.GetAwake)
	Do
	Do
	DisplayMoveOptions()
	MoveDirection = GetMove()
	ValidMove = CheckValidMove(MoveDirection)
	If ValidMove = False Then
	Console.WriteLine("That is not a valid move")
	End If
	Loop Until ValidMove
	If MoveDirection = "A" Then
	Dim ArrowDirection As Char = Player.GetArrowDirection
	If ArrowDirection = "N" And Monster.GetPosition.NoOfCellsEast = Player.GetPosition.NoOfCellsEast Then
	Console.WriteLine("You have shot the monster and it cannot stop you finding the flask")
	FlaskFound = True
	End If
	ElseIf MoveDirection <> "M" Then
		4	6	Answer written using VB.Net programming language.
				
				As was the case in 10.3, most of the correct functionality is here but there are errors which prevent it from working correctly under all the possible circumstances.
				
				The first error is that the conditions to check if the monster have been shot check (correctly) that the direction chosen was N and that the monster is in the same column in the grid as the player. However, there is no check that the monster is to the north of the player meaning that the monster will be shot by the arrow if it is directly to the north or directly to the south of the player's current position.
				
				The second error is that the check for the `MoveDirection` being equal to A is in the wrong place. The code provided means that if the player chooses to shoot an arrow then the player will be able to move again before the monster gets its turn (if it is awake). The question stated that "If the move chosen by the user is not M it then checks if the move chosen is A" – this is not what has been done here.
				
				Mark points 1 and 4 on the mark scheme have not been given; the other four marks have all been awarded.

```
        Cavern.PlaceItem(Player.GetPosition,
" ")
        Player.MakeMove(MoveDirection)
        Cavern.PlaceItem(Player.GetPosition,
"*")
        Cavern.Display(Monster.GetAwake)
        FlaskFound =
Player.CheckIfSameCell(Flask.GetPosition)
        If FlaskFound Then
          DisplayWonGameMessage()
        End If
        Eaten =
Monster.CheckIfSameCell(Player.GetPosition
)
        'This selection structure checks to
see if the player has triggered one of the
traps in the cavern
        If Not Monster.GetAwake And Not
FlaskFound And Not Eaten And
(Player.CheckIfSameCell(Trap1.GetPosition)
And Not Trap1.GetTriggered Or
Player.CheckIfSameCell(Trap2.GetPosition)
And Not Trap2.GetTriggered) Then
          Monster.ChangeSleepStatus()
          DisplayTrapMessage()
          Cavern.Display(Monster.GetAwake)
        End If
        If Monster.GetAwake And Not Eaten
And Not FlaskFound Then
          Count = 0
          Do
Cavern.PlaceItem(Monster.GetPosition, " ")
            Position = Monster.GetPosition
Monster.MakeMove(Player.GetPosition)
Cavern.PlaceItem(Monster.GetPosition, "M")
            If
Monster.CheckIfSameCell(Flask.GetPosition)
```

```
Then
            Flask.SetPosition(Position)
            Cavern.PlaceItem(Position,
"F")
          End If
          Eaten =
Monster.CheckIfSameCell(Player.GetPosition
)
          Console.WriteLine()
          Console.WriteLine("Press Enter
key to continue")
          Console.ReadLine()
          Cavern.Display(Monster.GetAwake)
          Count = Count + 1
        Loop Until Count = 2 Or Eaten
      End If
      If Eaten Then
        DisplayLostGameMessage()
      End If
    End If
  Loop Until Eaten Or FlaskFound Or
MoveDirection = "M"
End Sub
```

10.5		1	1	Even though there are errors in the program code, the code is mostly sensible so the testing mark is given here as the program code does work under the conditions of this test and the test has been carried out correctly.

10.6		1	1	Even though there are errors in the program code, the code is mostly sensible so the testing mark is given here as the program code does work under the conditions of this test and the test has been carried out correctly. The bottom part of the screen capture is truncated – but all the parts necessary to see that the test has been conducted and works correctly are visible on the screen capture provided.

Version 0.1
First published (07/10/2014)
Last updated (29/10/2014)