



**General Certificate of Education (A-level)
June 2012**

Computing

COMP1

(Specification 2510)

**Unit 1: Problem Solving, Programming, Data
Representation and Practical Exercise**

Final

Mark Scheme

Mark schemes are prepared by the Principal Examiner and considered, together with the relevant questions, by a panel of subject teachers. This mark scheme includes any amendments made at the standardisation events which all examiners participate in and is the scheme which was used by them in this examination. The standardisation process ensures that the mark scheme covers the candidates' responses to questions and that every examiner understands and applies it in the same correct way. As preparation for standardisation each examiner analyses a number of candidates' scripts: alternative answers not already covered by the mark scheme are discussed and legislated for. If, after the standardisation process, examiners encounter unusual answers which have not been raised they are required to refer these to the Principal Examiner.

It must be stressed that a mark scheme is a working document, in many cases further developed and expanded on the basis of candidates' reactions to a particular paper. Assumptions about future mark schemes on the basis of one year's document should be avoided; whilst the guiding principles of assessment remain constant, details will change, depending on the content of a particular examination paper.

Further copies of this Mark Scheme are available from: aqa.org.uk

Copyright © 2012 AQA and its licensors. All rights reserved.

Copyright

AQA retains the copyright on all its publications. However, registered centres for AQA are permitted to copy material from this booklet for their own internal use, with the following important exception: AQA cannot give permission to centres to photocopy any material that is acknowledged to a third party even for internal use within the centre.

Set and published by the Assessment and Qualifications Alliance.

To Examiners:

- When to award '0' (zero) when inputting marks on QMS and on scripts:** A mark of 0 should be awarded where a candidate has attempted a question but failed to write anything creditworthy. Insert a hyphen when a candidate has not attempted a question. By these two actions the Principal Examiner will be able to distinguish between the two (nothing creditworthy/unattempted) when analysing any statistics.
- This mark scheme contains the correct responses which we believe that candidates are most likely to give. Other valid responses are possible to some questions and should be credited. Examiners should refer off-mark scheme responses that they believe are creditworthy to a Team Leader.

The following annotation may be used in the mark scheme:

- ;
 - //
 - /
 - A
 - R
 - NE
 - I
 - DPT
- means a single mark
 - means alternative response
 - means an alternative word or sub-phrase
 - means acceptable creditworthy answer
 - means reject answer as not creditworthy
 - means not enough
 - means ignore
 - means 'Don't penalise twice'

No marks will be awarded for answers to testing questions where there is no evidence of programming code for the question(s) asked or where the screen captures provided by the candidate do not match what would be produced by the programming code.

Qu	Part	Marking Guidance	Marks																				
1	01	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th></th> <th></th> <th>Answer</th> <th>Carry</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>0</td> <td>1</td> </tr> </tbody> </table> <p style="text-align: right;">; ; ;</p> <p>A. 10 instead of 0 in the Answer column for the final row of the table</p>			Answer	Carry	0	0	0	0	0	1	1	0	1	0	1	0	1	1	0	1	3
		Answer	Carry																				
0	0	0	0																				
0	1	1	0																				
1	0	1	0																				
1	1	0	1																				
2	02	011 0010; R. If not 7 bits	1																				
	03	1011 0000 Mark as follows: Correct data bits; Correct parity bit for the candidate's data bits; R. If not 8 bits	2																				
	04	Error correction (not just error detection) (for single errors); Can detect when two errors have occurred in data transmission; Reduces the need for the retransmission of data; Decreases the likelihood of an undetected error // improved error																					

		detection; Can locate an error (not just detect that an error has occurred);	MAX 1
--	--	---	------------------

3	05	300; * 2; // 600;; NOTE: award 1 mark for doubling an incorrectly calculated highest frequency	2
---	----	---	----------

	06	Regular samples are taken (of the analogue signal); Samples are quantised // the height of each sample is approximated to an integer value // height of samples measured // amplitude/volume measured; Each integer value is encoded as a binary value // measurements are coded in a fixed number of bits; output the binary numbers as digital signals/voltage levels;	MAX 3
--	----	---	------------------

	07	Can (easily) synthesise musical notation from it; Can be played on different instruments; Can be (easily) transposed to a different key/pitch; Produces (relatively) small files; Easy to manipulate (the data); Allows for easy interface with electronic musical instruments; No data lost about a musical note;	MAX 1
--	----	--	------------------

	08	Length/duration (of note) // Note-on and Note-off; Instrument; Velocity//Speed; Volume//Amplitude; Timbre; Pedal effects; Channel; Instructions about how to recreate a sound; Aftertouch; Pitch bend; Note envelope; R. Note/key/pitch/frequency; A. Other sensible answers;	MAX 1
--	----	---	------------------

4	09	<table border="1" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th>Original State</th> <th>Input</th> <th>New State</th> </tr> </thead> <tbody> <tr> <td>S0</td> <td>10</td> <td>S10</td> </tr> <tr> <td>S0</td> <td>20</td> <td>S20</td> </tr> <tr> <td>S0</td> <td>50</td> <td>S50</td> </tr> <tr> <td>S0</td> <td>R</td> <td>S0</td> </tr> </tbody> </table> <p style="text-align: right;">; ; ; ; ;</p> <p><i>Note: order of completed rows not important</i></p>	Original State	Input	New State	S0	10	S10	S0	20	S20	S0	50	S50	S0	R	S0	3
Original State	Input	New State																
S0	10	S10																
S0	20	S20																
S0	50	S50																
S0	R	S0																

	10	20, 20, 10; R, R, 50;	
--	----	--------------------------	--

		10, 20, 20; 20, 50, 50; 20, R, 50;	MAX 4																																
5	11	(Each pixel) can be one of $4/2^2$ possible colours/values // Two bits are needed to represent the 4 possible bit patterns/colours/values // because there are 4/more than 2 colours in the image;	1																																
	12	<table border="1" style="margin-left: 20px;"> <tr> <td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td style="border: 2px solid black;">1</td><td style="border: 2px solid black;">0</td><td>1</td><td>1</td> </tr> </table> <p style="margin-left: 20px;">//</p> <table border="1" style="margin-left: 20px;"> <tr> <td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td style="border: 2px solid black;">0</td><td style="border: 2px solid black;">1</td><td>1</td><td>1</td> </tr> </table> <p>Mark as follows: 13th and 14th bits correct; Other bits correct;</p>	1	1	1	1	1	1	0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	0	0	0	0	1	1	0	1	1	1	2
1	1	1	1	1	1	0	0	0	0	1	1	1	0	1	1																				
1	1	1	1	1	1	0	0	0	0	1	1	0	1	1	1																				
	13	8*8 =64; * 2 = 128; ÷ 8 = 16; // 8*8*2÷ 8;;; 16;;; A. 128 <u>bits</u> as being worth 2 marks	3																																
	14	(Type of) shape // rectangle // square; Coordinates of corner/corners // position of a corner // top left coordinates; Identifier; Length of side(s) // width // height // coordinates of an opposing corner; Line colour // outer colour; Line width; Fill colour // inner colour; Angle of rotation; A. coordinates of midpoint/centre; A. radius/diameter A. circle/oval NE. Position/coordinates NE. Colour	MAX 3																																
	15	(For geometric images) less storage space/memory likely to be needed; NE. less space (For geometric images) will load faster from secondary storage; (For geometric images) will download faster; Can be scaled/resized without distortion; A. zoom Image can be (more easily) searched for particular objects; Can (more easily) manipulate individual objects in an image;	MAX 2																																
6	16	Correct variable declarations for <code>Bit</code> , <code>Answer</code> and <code>Column</code> ; I. additional variable declarations <code>Column</code> initialised correctly before the start of the loop; <code>Answer</code> initialised correctly before the start of the loop;																																	

		<p>While/Repeat loop, with syntax allowed by the programming language used, after the variable initialisations; and correct condition for the termination of the loop; R. For loop</p> <p>A. any While/Repeat loop with logic corresponding to that in flowchart (for a loop with a condition at the start accept ≥ 1 or > 0 but reject ≤ 0)</p> <p>Correct prompt "Enter bit value: "; followed by Bit assigned value entered by user; followed by Answer given new value; R. if incorrect value would be calculated followed by value of Column divided by 2; A. multiplying by 0.5 correct prompt and the assignment statements altering Bit, Answer and Column are all within the loop; After the loop - output message followed by value of Answer;</p> <p>I. Case of variable names, player names and output messages A. Minor typos in variable names and output messages I. spacing in prompts A. answers where formatting of decimal values is included e.g. <code>WriteLn('Decimal value is: ', Answer : 3)</code> A. initialisation of variables at declaration stage A. no brackets around <code>column * bit</code></p>	11
	17	<p>****SCREEN CAPTURE****</p> <p><i>Must match code from 16, including prompts on screen capture matching those in code</i></p> <p>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;</p>	3
	18	15;	1
	19	16 // twice as many // double;	1
	20	Design; A. Planning	1
	21	Implementation;	1
7	22	<p>ResetCavern; (all languages) // GetNewRandomPosition (Pascal only) // WriteWithMsg (VB6 only) // WriteLineWithMsg (VB6 only) // WriteLine (VB6 only) // WriteNoLine (VB6 only) // ReadLine (VB6 only); // SetUpTrapPostions (Python / Java only);</p> <p>R. if any additional code (including routine interface) R. if spelt incorrectly</p>	

		I. case	1
23	DisplayMenu // DisplayMoveOptions // DisplayWonGameMessage // DisplayTrapMessage // DisplayLostGameMessage // WriteWithMsg (VB6 only) // WriteLineWithMsg (VB6 only) // WriteLine (VB6 only) // WriteNoLine (VB6 only); A. DisplayCavern; R. if any additional code (including routine interface) R. if spelt incorrectly I. case	1	
24	Count1 // Count2 // Count; R. if any additional code R. if spelt incorrectly I. case	1	
25	Cavern // TrapPositions; R. if any additional code (including routine interface) R. if spelt incorrectly A. LoadedGameData.TrapPositions A. CurrentGameData.TrapPositions I. case	1	
26	When the value of the cell in the Cavern array // When the value of the cell to place the item in; Indicated by the Position variable; Contains a space // does not contain another item; R. is empty MAX 2 if <u>no</u> variable names used in description	3	
27	The number of times to repeat is <u>known</u> ; A. fixed	1	
28	Makes the program code easier to understand; Makes it easier to update the program; Makes it easier to change the number of traps (in the game);	MAX 1	
29	In text files all data is stored as strings/ASCII values/lines/characters // Text files use only byte values that display sensibly on a VDU // stores only values that can be opened and read in a text editor; Binary files store data using different data types; A. by example A. Binary files can only be correctly interpreted by application that created them	2	
30	Easier reuse of routines in other programs; Routine can be included in a library; Helps to make the program code more understandable; Ensures that the routine is self-contained // routine is independent of the rest of the program;		

		(Global variables use memory while a program is running) but local variables use memory for only part of the time a program is running; reduces possibility of undesirable side effects; Using global variables makes a program harder to debug;	MAX 2
	31	(If it was not then) <code>MonsterAwake</code> is set to the Boolean value returned by the second call to <code>CheckIfSameCell</code> ; this would overwrite any <code>True</code> value returned by the first call to <code>CheckIfSameCell</code> ; // Otherwise the monster would never wake up when the player triggers the first trap;; // Otherwise the monster would only wake up when the player triggers the second trap;;	2

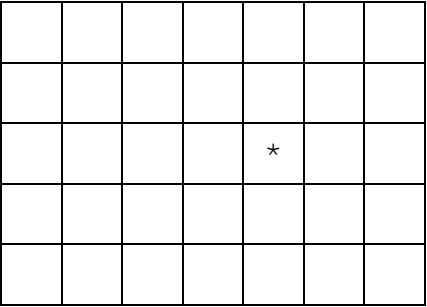
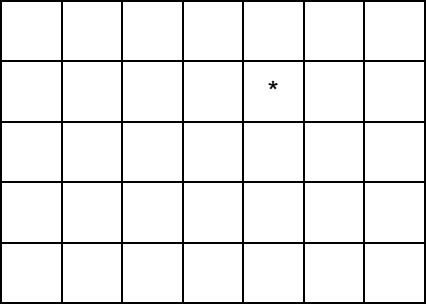
8	32	Appropriate option added; A. Any sensible prompt A. Prompt added anywhere in subroutine R. If prompt asks for character other than D	1
	33	Additional case statement for option D added correctly and all of the rest of the code added inside the correct option of the case statement; A. any character instead of D (except N, S, W, E) – only if matches prompt from 32 <code>NoOfCellsSouth</code> incremented within the correct option of the case statement; <code>NoOfCellsEast</code> incremented within the correct option of the case statement;	3
	34	Additional condition added to <code>IF</code> statement ; A. answers using an additional <code>IF</code> statement R. if value of 'D' will result in <code>False</code> being returned by function R. if function will always return <code>True</code>	1
	35	****SCREEN CAPTURE(S)**** <i>This is conditional on sensible code for 32, 33 and 34</i> Screen capture(s) showing modified menu shown to user and option 'D' selected; Screen capture(s) showing both original position of player in the cavern and the new position of the player in the cavern;	2

9	36	Selection structure with correct condition; Inside the selection structure there is code that will display the correct message on the screen; I. Capitalisation and minor typos in message R. different message Selection structure is in the correct place in the code;	3
	37	<code>If</code> statement with a correct condition;	

		<p>Correct logic and 2nd condition for <code>If</code> statement;</p> <p>Value of <code>False</code> returned correctly by the function if illegal north move is made;</p> <p>R. if a value of <code>False</code> will always be returned by the function</p> <p>R. if all north moves will return false</p> <p>R. if all moves when <code>PlayerPosition.NoOfCellsSouth</code> is in row 1 will return false</p> <p>Value of <code>True</code> returned correctly by the function if legal north move is made;</p> <p>A. Answers which combine all the checks for a valid move into one <code>If</code> statement</p> <p>I. missing option 'D' in code</p>	4
	38	<p>****SCREEN CAPTURE(S)****</p> <p><i>This is conditional on sensible code for 36 and correct code for 37</i></p> <p>Screen capture(s) showing correct cavern state with a player at the northern end of the cavern (top line), 'N' being entered at prompt, followed by correct error message being displayed;</p>	1
10	39	<p><code>NoOfMoves</code> is assigned the value 0 – before the first repetition structure in <code>PlayGame</code>;</p> <p>I. Case of variable names</p> <p>A. Minor typos in variable name</p> <p>A. assignment statement(s) in other subroutine(s) if correct functionality would be obtained</p> <p><code>NoOfMoves</code> incremented in any sensible place in the code inside the first selection structure in <code>PlayGame</code> subroutine;</p> <p>One correct message displayed with <code>NoOfMoves</code>;</p> <p>Second correct message displayed with <code>NoOfMoves</code>;</p> <p>Correct logic – both of the messages will be displayed only under the correct circumstances;</p> <p>A. minor typos in messages I. capitalisation & spacing in messages</p> <p>A. message displayed on more than one line</p> <p>A. more than one line of code used to display a message</p> <p>A. <code>NoOfMoves</code> declared as global</p> <p>I. <code>NoOfMoves</code> declaration not shown in PROGRAM SOURCE CODE</p>	5
	40	<p>****SCREEN CAPTURE(S)****</p> <p><i>This is conditional on sensible code for 39</i></p> <p>Screen capture(s) showing correct cavern state:</p>	

		<table border="1" data-bbox="370 210 799 510"> <tr><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></tr> <tr><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></tr> <tr><td></td><td></td><td></td><td>M</td><td></td><td>*</td><td></td></tr> </table> <p data-bbox="308 517 1198 618">followed by message "The number of moves you took to find the flask was 3";</p> <p data-bbox="308 658 1158 725">A. Different message – if it matches code in 39 and displays final value of NoOfMoves correctly</p> <p data-bbox="308 730 1059 797">R. If message "The number of moves that you survived ..." is also shown</p>																																M		*		1
			M		*																																	
41		<p data-bbox="308 837 711 869">****SCREEN CAPTURE(S)****</p> <p data-bbox="308 871 860 902"><i>This is conditional on sensible code for 39</i></p> <p data-bbox="308 936 935 967">Screen capture(s) showing correct cavern state:</p> <table border="1" data-bbox="370 976 799 1276"> <tr><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></tr> <tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td>M</td><td></td><td></td><td></td></tr> <tr><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr> </table> <p data-bbox="308 1308 1155 1375">followed by message "The number of moves that you survived in the cavern for was 2";</p> <p data-bbox="308 1415 1158 1482">A. Different message – if it matches code in 39 and displays final value of NoOfMoves correctly</p> <p data-bbox="308 1487 1190 1554">R. If message "The number of moves you took..." is also shown</p>																									M											1
			M																																			
11	42	<p data-bbox="308 1621 1107 1688">CalculateDistance subroutine created – with begin and end of subroutine;</p> <p data-bbox="308 1693 1070 1760">PlayerPosition and MonsterPosition passed as parameters to the CalculateDistance subroutine;</p> <p data-bbox="308 1765 788 1796">I. additional unnecessary parameters</p> <p data-bbox="308 1800 549 1832">R. global variables</p> <p data-bbox="308 1836 1134 1868">A. four integer values instead of two CellReference values</p> <p data-bbox="308 1872 1118 1939">R. passing by value for parameters of type CellReference (VB6 only)</p> <p data-bbox="308 1971 1187 2069">Integer value returned by subroutine either as parameter passed by reference or by function return value; R. global variable A. real value</p>																																				

	<p>Calculates difference between the <code>NoOfCellsEast</code> for the monster and the player; R. if the result can be a negative distance</p> <p>Calculates difference between the <code>NoOfCellsSouth</code> for the monster and the player; R. if the result can be a negative distance</p> <p>Calculates the total distance between the monster and the player; A. Incorrect values for differences in <code>NoOfCellsEast</code> and <code>NoOfCellsSouth</code> being added together</p> <p>Distance calculated is actually returned by the subroutine; A. use of global variable</p> <p>I. Case of identifiers A. Minor typos in identifiers I. Order of parameters in routine interface</p>	7
43	<p>Call to <code>CalculateDistance</code> subroutine; R. if parameter list does not match answer to 42</p> <p>Displays "Distance between monster and player: " in correct place; A. any place in code after call to <code>DisplayMoveOptions</code> and before call to <code>MakeMove</code> A. minor typos in prompt I. capitalisation</p> <p>Displays the calculated distance; R. if no evidence of any calculation for the distance R. if distance is displayed before call to <code>CalculateDistance</code> subroutine R. if distance returned by <code>CalculateDistance</code> stored in a global variable R. if distance calculated in part 42 would not actually be displayed e.g. program would not compile/run A. use of temporary variable to store the value returned by <code>CalculateDistance</code> with contents of temporary variable then displayed using output message</p> <p>I. Case of identifiers and output messages A. Minor typos in output messages I. spacing in output messages</p>	3

<p>44</p>	<p>****SCREEN CAPTURE(S)**** <i>This is conditional on sensible code for 42 and/or 43</i></p> <p>Player shown in the cell 3 south and 5 east of the northwest corner AND "Distance between monster and player: 3" shown;</p>  <p>I. monster symbol (M) displayed in the cavern</p>	<p>1</p>
<p>45</p>	<p>****SCREEN CAPTURE(S)**** <i>This is conditional on sensible code for 42 and/or 43</i></p> <p>Player shown in the cell 2 south and 5 east of the northwest corner AND "Distance between monster and player: 2" shown;</p>  <p>I. monster symbol (M) displayed in the cavern</p>	<p>1</p>

46	<p>****SCREEN CAPTURE(S)**** <i>This is conditional on sensible code for 42 and/or 43</i></p> <p>Player shown in the cell 2 south and 3 east of the northwest corner AND "Distance between monster and player: 2" shown;</p> <div style="display: flex; align-items: center; justify-content: center;"> <div style="margin-right: 10px;"> <p>I. monster displayed in the</p> </div> <table border="1" style="border-collapse: collapse; text-align: center;"> <tr><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></tr> <tr><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></tr> <tr><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td><td> </td></tr> </table> <div style="margin-left: 10px;"> <p>symbol (M) cavern</p> </div> </div>										*																										1
		*																																			

PASCAL Mark Scheme

Qu	Part	Marking Guidance	Marks
6	16	<pre> 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. </pre>	11
8	32	<pre> Procedure DisplayMoveOptions; Begin Writeln; Writeln('Enter N to move NORTH'); Writeln('Enter E to move EAST'); Writeln('Enter S to move SOUTH'); Writeln('Enter W to move WEST'); Writeln('Enter D to move SOUTHEAST'); Writeln('Enter M to return to the Main Menu'); Writeln; End; </pre>	1
	33	<pre> Case Direction Of 'N' : PlayerPosition.NoOfCellsSouth := PlayerPosition.NoOfCellsSouth - 1; 'S' : PlayerPosition.NoOfCellsSouth := PlayerPosition.NoOfCellsSouth + 1; 'W' : PlayerPosition.NoOfCellsEast := PlayerPosition.NoOfCellsEast - 1; 'E' : PlayerPosition.NoOfCellsEast := PlayerPosition.NoOfCellsEast + 1; 'D' : Begin PlayerPosition.NoOfCellsSouth := PlayerPosition.NoOfCellsSouth + 1; PlayerPosition.NoOfCellsEast := PlayerPosition.NoOfCellsEast + 1; End; End; </pre>	3

	34	<pre>ValidMove := True; If Not (Direction In ['N','S','W','E','D','M']) Then ValidMove := False; CheckValidMove := ValidMove;</pre>	1
9	36	<pre>Repeat DisplayMoveOptions; MoveDirection := GetMove; ValidMove := CheckValidMove(PlayerPosition, MoveDirection); If Not ValidMove Then Writeln('That is not a valid move, please try again'); Until ValidMove; Alternative answer If ValidMove = False...</pre>	3
	37	<pre>ValidMove := True; If Not (Direction In ['N','S','W','E','D','M']) Then ValidMove := False; If (PlayerPosition.NoOfCellsSouth = 1) And (Direction = 'N') Then ValidMove := False; CheckValidMove := ValidMove; Alternative answer If ValidMove And (Direction = 'N') Then ValidMove := ValidMove And (PlayerPosition <> 1);</pre>	4
10	39	<pre>Eaten:= False; FlaskFound := False; DisplayCavern(Cavern, MonsterAwake); NoOfMoves := 0; Repeat ... If MoveDirection <> 'M' Then Begin MakeMove(Cavern, MoveDirection, PlayerPosition); NoOfMoves := NoOfMoves + 1; DisplayCavern(Cavern, MonsterAwake); ... If FlaskFound Then Begin DisplayWonGameMessage;</pre>	

	<pre> Writeln('The number of moves you took to find the flask was ',NoOfMoves); End; ... If Eaten Then Begin DisplayLostGameMessage; Writeln('The number of moves you survived in the cavern for was ', NoOfMoves); End; </pre> <p>Alternative answer Until Eaten Or FlaskFound Or (MoveDirection = 'M'); If Eaten Then Writeln('The number of moves that you survived in the cavern for was ', NoOfMoves); If FlaskFound Then Writeln('The number of moves you took to find the flask was ', NoOfMoves);</p> <p>Alternative answer If FlaskFound Then DisplayWonGameMessage(NoOfMoves) ; ... If Eaten Then DisplayLostGameMessage(NoOfMoves) ;</p> <p>together with modified DisplayWonGameMessage to include additional output message (l. missing parameter if NoOfMoves declared as global)</p> <pre> Procedure DisplayWonGameMessage(NoOfMoves : Integer) ; Begin Writeln('Well done! You have found the flask containing the Styxian potion. '); Writeln('You have won the game of MONSTER! '); Writeln('The number of moves you took to find the flask was ',NoOfMoves); Writeln; End </pre> <p>and modified DisplayLostGameMessage to include additional output message (l. missing parameter if NoOfMoves declared as global)</p> <pre> Procedure DisplayLostGameMessage(NoOfMoves : Integer) ; Begin Writeln('ARGHHHHHH! The monster has eaten you. GAME OVER. '); </pre>	
--	---	--

		<pre> Writeln('Maybe you will have better luck next time you play MONSTER!'); Writeln('The number of moves you survived in the cavern for was ', NoOfMoves); Writeln; End; </pre>	5
11	42	<pre> Function CalculateDistance(PlayerPosition, MonsterPosition : TCellReference) : Integer; Var Distance : Integer; Begin If PlayerPosition.NoOfCellsEast > MonsterPosition.NoOfCellsEast Then Distance := PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast Else Distance := MonsterPosition.NoOfCellsEast - PlayerPosition.NoOfCellsEast; If PlayerPosition.NoOfCellsSouth > MonsterPosition.NoOfCellsSouth Then Distance := Distance + PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth Else Distance := Distance + MonsterPosition.NoOfCellsSouth - PlayerPosition.NoOfCellsSouth; CalculateDistance := Distance; End; </pre> <p>Alternative answer Distance := Abs(PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast) + Abs(PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth));</p> <p>Alternative answer Distance := Trunc(Sqrt(Sqr(PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast)) + Sqrt(Sqr(PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth)));</p> <p>Alternative answer Distance := Round(Sqrt(Sqr(PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast)) + Sqrt(Sqr(PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth)));</p> <p>Alternative answer</p>	

	<pre> Distance2 : Integer; ... Distance := PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast; If Distance < 0 Then Distance := Distance * -1; Distance2 := PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth; If Distance2 < 0 Then Distance2 := Distance2 * -1; Distance := Distance + Distance2; </pre>	7
43	<pre> DisplayMoveOptions; Writeln('Distance between monster and player: ', CalculateDistance(PlayerPosition, MonsterPosition)); </pre>	3

VB.NET Mark Scheme

Qu	Part	Marking Guidance	Marks
6	16	<pre> Sub Main() Dim Answer As Integer Dim Column As Integer Dim Bit As Integer Answer = 0 Column = 8 Do Console.WriteLine("Enter bit value: ") Bit = Console.ReadLine Answer = Answer + (Column * Bit) Column = Column / 2 Loop Until Column < 1 Console.WriteLine("Decimal value is: " & Answer) Console.ReadLine() End Sub Alternative Answer Column = Column \ 2 </pre>	11
8	32	<pre> 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 D to move SOUTHEAST") Console.WriteLine("Enter M to return to the Main Menu") Console.WriteLine() End Sub </pre>	1
	33	<pre> Case "E" PlayerPosition.NoOfCellsEast = PlayerPosition.NoOfCellsEast + 1 Case "D" PlayerPosition.NoOfCellsSouth = PlayerPosition.NoOfCellsSouth + 1 PlayerPosition.NoOfCellsEast = PlayerPosition.NoOfCellsEast + 1 </pre>	3
	34	<pre> ValidMove = True If Not (Direction = "N" Or Direction = "S" Or Direction = "W" Or Direction = "E" Or Direction = "M Or Direction = "D") Then ValidMove = False End If CheckValidMove = ValidMove </pre>	1

9	36	<pre> Do DisplayMoveOptions() MoveDirection = GetMove() ValidMove = CheckValidMove(PlayerPosition, MoveDirection) If Not ValidMove Then Console.WriteLine("That is not a valid move, please try again") End If Loop Until ValidMove </pre>	3
	37	<pre> If Not (Direction = "N" Or Direction = "S" Or Direction = "W" Or Direction = "E" Or Direction = "D" Or Direction = "M") Then ValidMove = False End If If PlayerPosition.NoOfCellsSouth = 1 And Direction = "N" Then ValidMove = False End If CheckValidMove = ValidMove Alternative answer If Not (Direction = "N" Or Direction = "S" Or Direction = "W" Or Direction = "E" Or Direction = "M") Then ValidMove = False End If If ValidMove And (Direction = "N") Then ValidMove = (ValidMove And (PlayerPosition.NoOfCellsSouth <> 1)) End If </pre>	4
10	39	<pre> Dim ValidMove As Boolean Eaten = False FlaskFound = False DisplayCavern(Cavern, MonsterAwake) NoOfMoves = 0 Do ... If MoveDirection <> "M" Then MakeMove(Cavern, MoveDirection, PlayerPosition) NoOfMoves = NoOfMoves + 1 DisplayCavern(Cavern, MonsterAwake) ... If FlaskFound Then DisplayWonGameMessage() Console.WriteLine("The number of moves you took to find the flask was " & NoOfMoves) End If </pre>	

...

```
If Eaten Then
    DisplayLostGameMessage()
    Console.WriteLine("The number of moves that you
survived in the cavern for was " & NoOfMoves)
End If
```

...

Alternative answer

```
Loop Until Eaten Or FlaskFound Or MoveDirection =
"M"
```

```
If Eaten Then
    Console.WriteLine("The number of moves that you
survived in the cavern for was " & NoOfMoves)
End If
```

```
If FlaskFound Then
    Console.WriteLine("The number of moves you took
to find the flask was " & NoOfMoves)
End If
```

Alternative answer

```
If FlaskFound Then
    DisplayWonGameMessage(NoOfMoves)
End If
```

...

```
If Eaten Then
    DisplayLostGameMessage(NoOfMoves)
End If
```

together with modified DisplayWonGameMessage to include additional output message (i. missing parameter if NoOfMoves declared as global)

```
Sub DisplayWonGameMessage(ByVal NoOfMoves As
Integer)
    Console.WriteLine("Well done! You have found the
flask containing the Styxian potion.")
    Console.WriteLine("You have won the game of
MONSTER!")
    Console.WriteLine("The number of moves you took
to find the flask was " & NoOfMoves)
    Console.WriteLine()
End Sub
```

and modified DisplayLostGameMessage to include additional output message (i. missing parameter if NoOfMoves declared as global)

```
Sub DisplayLostGameMessage(ByVal NoOfMoves As
Integer)
    Console.WriteLine("ARGHHHHHH! The monster has
```

	<pre>eaten you. GAME OVER.") Console.WriteLine("Maybe you will have better luck next time you play MONSTER!") Console.WriteLine("The number of moves you survived in the cavern for was " & NoOfMoves); Console.WriteLine() End Sub</pre>	5
--	---	----------

11	42	<pre>Function CalculateDistance(ByVal PlayerPosition As CellReference, ByVal MonsterPosition As CellReference) As Integer Dim Distance As Integer If PlayerPosition.NoOfCellsEast > MonsterPosition.NoOfCellsEast Then Distance = PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast Else Distance = MonsterPosition.NoOfCellsEast - PlayerPosition.NoOfCellsEast End If If PlayerPosition.NoOfCellsSouth > MonsterPosition.NoOfCellsSouth Then Distance = Distance + PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth Else Distance = Distance + MonsterPosition.NoOfCellsSouth - PlayerPosition.NoOfCellsSouth End If CalculateDistance = Distance End Function</pre> <p>Alternative answer Distance = System.Math.Abs (PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast) + System.Math.Abs (PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth)</p> <p>A. this alternative answer if System.Math included A. give benefit of doubt for this answer if no evidence of System.Math included</p> <p>Alternative answer Distance = (((PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast) ^ 2) ^ 0.5) + (((PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth) ^ 2) ^ 0.5)</p>	
----	----	--	--

	<p>Alternative answer</p> <pre>Dim Distance2 As Integer ... Distance = PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast If Distance < 0 Then Distance = Distance * -1 End If Distance2 = PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth If Distance2 < 0 Then Distance2 = Distance2 * -1 End If Distance = Distance + Distance2</pre>	7
43	<pre>DisplayMoveOptions() Console.WriteLine("Distance between monster and player: " & CalculateDistance(PlayerPosition, MonsterPosition))</pre>	3

VB6 Mark Scheme

Qu	Part	Marking Guidance	Marks
6	16	<pre>Private Sub Form_Load() Dim Answer As Integer Dim Column As Integer Dim Bit As Integer Answer = 0 Column = 8 Do Bit = InputBox("Enter bit value: ") Answer = Answer + (Column * Bit) Column = Column / 2 Loop Until Column < 1 MsgBox ("Decimal value is: " & Answer) End Sub</pre> <p>Alternative Answer Column = Column \ 2</p>	11
8	32	<pre>Private Sub DisplayMoveOptions() WriteLine ("") WriteLine ("Enter N to move NORTH") WriteLine ("Enter E to move EAST") WriteLine ("Enter S to move SOUTH") WriteLine ("Enter W to move WEST") WriteLine ("Enter D to move SOUTHEAST") WriteLine ("Enter M to return to the Main Menu") WriteLine ("") End Sub</pre> <p>A. Text1.Text = Text1.Text & "Enter D to move SOUTHEAST "</p>	1
	33	<pre>Case "E" PlayerPosition.NoOfCellsEast = PlayerPosition.NoOfCellsEast + 1 Case "D" PlayerPosition.NoOfCellsSouth = PlayerPosition.NoOfCellsSouth + 1 PlayerPosition.NoOfCellsEast = PlayerPosition.NoOfCellsEast + 1</pre>	3
	34	<pre>ValidMove = True If Not (Direction = "N" Or Direction = "S" Or Direction = "W" Or Direction = "E" Or Direction = "M" Or Direction = "D") Then ValidMove = False End If CheckValidMove = ValidMove</pre>	1

9	36	<pre> Do Call DisplayMoveOptions() MoveDirection = GetMove() ValidMove = CheckValidMove(PlayerPosition, MoveDirection) If Not ValidMove Then WriteLine("That is not a valid move, please try again") End If Loop Until ValidMove A. Text1.Text = Text1.Text & "That is not a valid move, please try again" A. WriteLineWithMsg </pre>	3
	37	<pre> If Not (Direction = "N" Or Direction = "S" Or Direction = "W" Or Direction = "E" Or Direction = "D" Or Direction = "M") Then ValidMove = False End If If PlayerPosition.NoOfCellsSouth = 1 And Direction = "N" Then ValidMove = False End If CheckValidMove = ValidMove Alternative answer If Not (Direction = "N" Or Direction = "S" Or Direction = "W" Or Direction = "E" Or Direction = "M") Then ValidMove = False End If If ValidMove And (Direction = "N") Then ValidMove = (ValidMove And (PlayerPosition.NoOfCellsSouth <> 1)) End If </pre>	4
10	39	<pre> Dim ValidMove As Boolean Eaten = False FlaskFound = False Call DisplayCavern(Cavern, MonsterAwake) NoOfMoves = 0 Do ... If MoveDirection <> "M" Then Call MakeMove(Cavern, MoveDirection, PlayerPosition) NoOfMoves = NoOfMoves + 1 Call DisplayCavern(Cavern, MonsterAwake) ... </pre>	

	<pre> If FlaskFound Then Call DisplayWonGameMessage() WriteLine("The number of moves you took to find the flask was " & NoOfMoves) End If ... If Eaten Then Call DisplayLostGameMessage() WriteLine("The number of moves that you survived in the cavern for was " & NoOfMoves) End If ... Alternative answer Loop Until Eaten Or FlaskFound Or MoveDirection = "M" If Eaten Then WriteLine("The number of moves that you survived in the cavern for was " & NoOfMoves) End If If FlaskFound Then WriteLine("The number of moves you took to find the flask was " & NoOfMoves) End If Alternative answer If FlaskFound Then DisplayWonGameMessage(NoOfMoves) End If ... If Eaten Then DisplayLostGameMessage(NoOfMoves) End If together with modified DisplayWonGameMessage to include additional output message (l. missing parameter if NoOfMoves declared as global) Sub DisplayWonGameMessage(ByVal NoOfMoves As Integer) WriteLine("Well done! You have found the flask containing the Styxian potion.") WriteLine("You have won the game of MONSTER!") Writeline("The number of moves you took to find the flask was " & NoOfMoves); WriteLine("") End Sub and modified DisplayLostGameMessage to include additional output message (l. missing parameter if NoOfMoves declared as global) </pre>	
--	---	--

	<pre> Sub DisplayLostGameMessage (ByVal NoOfMoves As Integer) WriteLine("ARGHHHHHH! The monster has eaten you. GAME OVER.") WriteLine("Maybe you will have better luck next time you play MONSTER!") WriteLine("The number of moves you survived in the cavern for was " & NoOfMoves); WriteLine("") End Sub A. Text1.Text = Text1.Text & "The number of moves that you survived in the cavern for was " A. Text1.Text = Text1.Text & "The number of moves you took to find the flask was " A. WriteLineWithMsg </pre>	5
--	---	----------

11	<p>42</p> <pre> Private Function CalculateDistance (ByRef PlayerPosition As CellReference, ByRef MonsterPosition As CellReference) As Integer Dim Distance As Integer If PlayerPosition.NoOfCellsEast > MonsterPosition.NoOfCellsEast Then Distance = PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast Else Distance = MonsterPosition.NoOfCellsEast - PlayerPosition.NoOfCellsEast End If If PlayerPosition.NoOfCellsSouth > MonsterPosition.NoOfCellsSouth Then Distance = Distance + PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth Else Distance = Distance + MonsterPosition.NoOfCellsSouth - PlayerPosition.NoOfCellsSouth End If CalculateDistance = Distance End Function Alternative answer Distance = (((PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast) ^ 2) ^ 0.5) + (((PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth) ^ 2) ^ 0.5) </pre>	
----	---	--

	<p>Alternative answer Distance = Abs(PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast) + Abs(PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth)</p> <p>Alternative answer Dim Distance2 As Integer ... Distance = PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast If Distance < 0 Then Distance = Distance * -1 End If Distance2 = PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth If Distance2 < 0 Then Distance2 = Distance2 * -1 End If Distance = Distance + Distance2</p>	7
43	<p>DisplayMoveOptions() WriteLine("Distance between monster and player: " & CalculateDistance(PlayerPosition, MonsterPosition))</p> <p>A. Text1.Text = Text1.Text & "Distance between monster and player: " & CalculateDistance(PlayerPosition, MonsterPosition) A. WriteLineWithMsg</p>	3

JAVA Mark Scheme

Qu	Part	Marking Guidance	Marks
6	16	<pre> 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(); } } </pre>	11
8	32	<pre> void displayMoveOptions() { console.println(); console.println("Enter N to move NORTH"); console.println("Enter E to move EAST"); console.println("Enter S to move SOUTH"); console.println("Enter W to move WEST"); console.println("Enter D to move SOUTHEAST"); console.println("Enter M to return to the Main Menu"); console.println(); } </pre>	1
	33	<pre> switch (direction) { case 'N': playerPosition.noOfCellsSouth--; break; case 'S': playerPosition.noOfCellsSouth++; break; case 'W': playerPosition.noOfCellsEast--; break; case 'E': playerPosition.noOfCellsEast++; break; } </pre>	

		<pre> case 'D': playerPosition.noOfCellsSouth++; playerPosition.noOfCellsEast++; break; } </pre>	3
	34	<pre> validMove = true; if (!(direction == 'N' direction == 'S' direction == 'W' direction == 'E' direction == 'D' direction == 'M')) { validMove = false; } return validMove; </pre>	1
9	36	<pre> do { displayMoveOptions(); moveDirection = getMove(); validMove = checkValidMove(playerPosition, moveDirection); if (!validMove) { console.println("That is not a valid move, please try again"); } } while (!validMove); Alternative answer if (validMove == false) </pre>	3
	37	<pre> validMove = true; if (!(direction == 'N' direction == 'S' direction == 'W' direction == 'E' direction == 'D' direction == 'M')) { validMove = false; } if (validMove && direction == 'N') { validMove = validMove && (playerPosition.noOfCellsSouth != 1); } return validMove; Alternative answer if (playerPosition.noOfCellsSouth == 1 && direction == 'N') { validMove = false; } </pre>	4
10	39	<pre> eaten = false; flaskFound = false; displayCavern(cavern, monsterAwake); noOfMoves = 0; </pre>	

```

do {
    ...
    if (moveDirection != 'M') {
        makeMove(cavern, moveDirection,
playerPosition);
        noOfMoves++;
        displayCavern(cavern, monsterAwake);
        flaskFound = checkIfSameCell(playerPosition,
flaskPosition);
        if (flaskFound) {
            displayWonGameMessage();
            console.println("The number of moves you took
to find the flask was " + noOfMoves);
        }
        ...

        if (eaten) {
            displayLostGameMessage();
            console.println("The number of moves you
survived in the " + "cavern for was " +
noOfMoves);
        }
    }
}

```

Alternative answer

```

} while (!(eaten || flaskFound || moveDirection ==
'M'));
if (flaskFound) {
    console.println("The number of moves you took to
find the flask was " + noOfMoves);
}
if (eaten) {
    console.println("The number of moves you
survived in the " + "cavern for was " +
noOfMoves);
}

```

Alternative answer

```

eaten = false;
flaskFound = false;
displayCavern(cavern, monsterAwake);
noOfMoves = 0;
do {
    ...
    if (moveDirection != 'M') {
        makeMove(cavern, moveDirection,
playerPosition);
        noOfMoves++;
        displayCavern(cavern, monsterAwake);
        ...
    }
}

```

together with modified displayLostGameMessage and displayWonGameMessage to include additional output message (I.

	<p>missing parameter if NoOfMoves declared as global)</p> <pre> void displayWonGameMessage(int noOfMoves){ console.println("ARGHHHHHH! The monster has eaten you. GAME OVER."); console.println("Maybe you will have better luck next time you play MONSTER!"); console.println("The number of moves you survived in the cavern was " + noOfMoves); console.println(); } void displayWonGameMessage(int noOfMoves){ console.println("Well done! You have found the flask containing the Styxian potion."); console.println("You have won the game of MONSTER!"); console.println("The number of moves you took to find the flask was " + noOfMoves); } </pre>	5
--	--	----------

11	<p style="text-align: center;">42</p> <pre> int calculateDistance(CellReference playerPosition, CellReference monsterPosition) { int distance; if(playerPosition.noOfCellsEast>monsterPosition.no OfCellsEast){ distance=playerPosition.noOfCellsEast- monsterPosition.noOfCellsEast; } else{ distance=monsterPosition.noOfCellsEast- playerPosition.noOfCellsEast; } if(playerPosition.noOfCellsSouth>monsterPosition.n oOfCellsSouth){ distance=distance+playerPosition.noOfCellsSouth- monsterPosition.noOfCellsSouth; }else{ distance=distance+monsterPosition.noOfCellsSouth- playerPosition.noOfCellsSouth; } return distance; } Alternative Answer int calculateDistance(CellReference playerPosition, CellReference monsterPosition) { </pre>	
----	---	--

	<pre> int distance; distance = Math.abs(playerPosition.noOfCellsSouth - monsterPosition.noOfCellsSouth); distance += Math.abs(playerPosition.noOfCellsEast - monsterPosition.noOfCellsEast); return distance; } </pre> <p>Alternative Answer</p> <pre> distance=(int)Math.sqrt(Math.pow((double)(playerPo sition.noOfCellsSouth - monsterPosition.noOfCellsSouth), 2)) +(int)Math.sqrt(Math.pow((double)(playerPosition.n oOfCellsEast - monsterPosition.noOfCellsEast), 2)); </pre> <p>Alternative Answer</p> <pre> distance=(int)Math.round(Math.sqrt(Math.pow((doubl e)(playerPosition.noOfCellsSouth - monsterPosition.noOfCellsSouth), 2)) +Math.sqrt(Math.pow((double)(playerPosition.noOfCe llsEast - monsterPosition.noOfCellsEast), 2))); </pre> <p>Alternative answer</p> <pre> int distance2; ... distance = playerPosition.noOfCellsEast - monsterPosition.noOfCellsEast; if (distance < 0) { distance = distance * -1; } distance2 = playerPosition.noOfCellsSouth - monsterPosition.noOfCellsSouth; if (distance2 < 0) { distance2 = distance2 * -1; } distance = distance + distance2; </pre>	7
43	<pre> displayMoveOptions(); console.println("Distance between monster and player: " + calculateDistance(playerPosition, monsterPosition)); </pre>	3

PYTHON Mark Scheme

Qu	Part	Marking Guidance	Marks
6	16	<p># Section B Q6 Python 2.6 Answer = 0 Bit = 0 Column = 8</p> <pre>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)</pre> <p># Section B Q6 Python 3.1 Answer = 0 Bit = 0 Column = 8</p> <pre>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: {}".format(Answer))</pre> <p>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</p>	11
8	32	<p>Python 2 <pre>def DisplayMoveOptions(): print '' print 'Enter N to move NORTH' print 'Enter E to move EAST' print 'Enter S to move SOUTH' print 'Enter W to move WEST' print 'Enter D to move SOUTHEAST' print 'Enter M to return to the Main Menu' print ''</pre></p> <p>Python 3 <pre>def DisplayMoveOptions(): print () print ('Enter N to move NORTH') print ('Enter E to move EAST') print ('Enter S to move SOUTH')</pre></p>	

		<pre>print ('Enter W to move WEST') print ('Enter D to move SOUTHEAST') print ('Enter M to return to the Main Menu') print ()</pre>	1
	33	<pre>elif Direction == 'E': PlayerPosition.NoOfCellsEast += 1 elif Direction == 'D': PlayerPosition.NoOfCellsSouth += 1 PlayerPosition.NoOfCellsEast += 1</pre>	3
	34	<pre>def CheckValidMove(PlayerPosition,Direction): ValidMove = True if not (Direction in ['N','S','W','E','D','M']): ValidMove = False return ValidMove</pre>	1
9	36	<pre>while not ValidMove: DisplayMoveOptions() MoveDirection = GetMove() ValidMove = CheckValidMove(PlayerPosition, MoveDirection) if not ValidMove: # Python 2: print 'That is not a valid move, please try again' # Python 3: print('That is not a valid move, please try again') Alternative answer if ValidMove = False...</pre>	3
	37	<pre>def CheckValidMove(PlayerPosition,Direction): ValidMove = True if not (Direction in ['N','S','W','E','D','M']): ValidMove = False if (PlayerPosition.NoOfCellsSouth == 1) and (Direction == 'N'): ValidMove = False return ValidMove Alternative answer if not (Direction in ['N','S','W','E','D','M']): ValidMove = False if ValidMove and (Direction == 'N'): ValidMove = (ValidMove and (PlayerPosition. NoOfCellsSouth != 1))</pre>	4

10	39	<pre> Eaten = False FlaskFound = False MoveDirection = '' DisplayCavern(Cavern, MonsterAwake) NoOfMoves = 0 while not (Eaten or FlaskFound or (MoveDirection == 'M')): ValidMove = False while not ValidMove: DisplayMoveOptions() MoveDirection = GetMove() ValidMove = CheckValidMove(PlayerPosition, MoveDirection) if not ValidMove: print 'That is not a valid move, please try again' if MoveDirection != 'M': MakeMove(Cavern, MoveDirection, PlayerPosition) NoOfMoves += 1 DisplayCavern(Cavern, MonsterAwake) ... if FlaskFound: DisplayWonGameMessage() # Python 2: print 'The number of moves you took to find the flask was', NoOfMoves # Alternative answer: print 'The number of moves you took to find the flask was ' + str(NoOfMoves) # Python 3: print('The number of moves you took to find the flask was ' + str(NoOfMoves) # Alternative answer: print('The number of moves you took to find the flask was {0}'.format(NoOfMoves)) #Py3 ... if Eaten: DisplayLostGameMessage() # Python 2: print 'The number of moves that you survived in the cavern for was', NoOfMoves # Alternative answer: print 'The number of moves that you survived in the cavern for was ' + str(NoOfMoves) # Python 3: print('The number of moves that you survived in the cavern for was ' + str(NoOfMoves)) # Alternative answer: print('The number of moves that you survived in the cavern for was {0}'.format(NoOfMoves)) </pre>	
----	----	--	--

Alternative Answer

```
# Python 2
if Eaten:
    print 'The number of moves that you survived in the
cavern for was', NoOfMoves
else:
    print 'The number of moves you took to find the
flask was', NoOfMoves
```

```
# Python 3
if Eaten:
    print('The number of moves that you survived in the
cavern for was' + str(NoOfMoves))
else:
    print('The number of moves you took to find the
flask was' + str(NoOfMoves))
```

A. .format(NoOfMoves)

Alternative answer

```
if FlaskFound:
    DisplayWonGameMessage(NoOfMoves)
...
if Eaten:
    DisplayLostGameMessage(NoOfMoves)
```

together with modified displayLostGameMessage and displayWonGameMessage to include additional output message (l. missing parameter if NoOfMoves declared as global)

Python 2

```
def DisplayWonGameMessage(NoOfMoves):
    print 'Well Done! You have found the flask
containing the Styxian potion.'
    print 'You have won the game of MONSTER!'
    print 'The number of moves you took to find the
flask was ', NoOfMoves
```

```
def DisplayLostGameMessage(NoOfMoves):
    print 'ARGHHHHHH! The monster has eaten you. GAME
OVER.'
    print 'Maybe you will have better luck the next
time you play MONSTER!'
    print 'The number of moves that you survived in the
cavern for was', NoOfMoves
```

Python 3

```
def DisplayWonGameMessage(NoOfMoves):
    print('Well Done! You have found the flask
containing the Styxian potion.')
```

		<pre> print('You have won the game of MONSTER!') print('The number of moves you took to find the flask was' + str(NoOfMoves)) def DisplayLostGameMessage(NoOfMoves): print('ARGHHHHH! The monster has eaten you. GAME OVER.') print('Maybe you will have better luck the next time you play MONSTER!') print('The number of moves that you survived in the cavern for was'+ str(NoOfMoves)) </pre>	5
11	42	<pre> def CalculateDistance(PlayerPosition, MonsterPosition): if PlayerPosition.NoOfCellsEast > MonsterPosition.NoOfCellsEast: Distance = PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast else: Distance = MonsterPosition.NoOfCellsEast - PlayerPosition.NoOfCellsEast if PlayerPosition.NoOfCellsSouth > MonsterPosition.NoOfCellsSouth: Distance = Distance + PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth else: Distance = Distance + MonsterPosition.NoOfCellsSouth - PlayerPosition.NoOfCellsSouth return Distance Alternative Answer Distance = abs(PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast) + abs(PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth) Alternative Answer return abs(PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast) + abs(PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth) Alternative Answer import math Distance = math.trunc(math.sqrt(pow((PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast),2)) + math.sqrt(pow((PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth),2))) </pre>	

	<p>Alternative Answer import math Distance = round(math.sqrt((PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast)**2) + math.sqrt((PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth)**2))</p> <p>Alternative Answer Distance = PlayerPosition.NoOfCellsEast - MonsterPosition.NoOfCellsEast if Distance < 0: Distance = Distance * -1 Distance2 = PlayerPosition.NoOfCellsSouth - MonsterPosition.NoOfCellsSouth if Distance2 < 0: Distance2 = Distance2 * -1 Distance = Distance + Distance2</p>	7
43	<p># Python 2: DisplayMoveOptions() print 'Distance to monster:', CalculateDistance(PlayerPosition, MonsterPosition)</p> <p># Alternative answer: DisplayMoveOptions() print 'Distance to monster:' + str(CalculateDistance(PlayerPosition, MonsterPosition))</p> <p># Python 3: DisplayMoveOptions() print('Distance to monster:' + str(CalculateDistance(PlayerPosition, MonsterPosition))</p>	3

UMS conversion calculator www.aqa.org.uk/umconversion