

---

**COMPUTER SCIENCE**

**9608/42**

Paper 4 Written Paper

**May/June 2017**

MARK SCHEME

Maximum Mark: 75

---

**Published**

This mark scheme is published as an aid to teachers and candidates, to indicate the requirements of the examination. It shows the basis on which Examiners were instructed to award marks. It does not indicate the details of the discussions that took place at an Examiners' meeting before marking began, which would have considered the acceptability of alternative answers.

Mark schemes should be read in conjunction with the question paper and the Principal Examiner Report for Teachers.

Cambridge will not enter into discussions about these mark schemes.

Cambridge is publishing the mark schemes for the May/June 2017 series for most Cambridge IGCSE<sup>®</sup>, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

© IGCSE is a registered trademark.

---

This document consists of **19** printed pages.

Question	Answer				Marks																																																																	
1(a)	<table border="1"> <thead> <tr> <th data-bbox="349 201 528 284">Label</th> <th data-bbox="535 201 618 284">Op code</th> <th data-bbox="624 201 792 284">Operand</th> <th data-bbox="799 201 1554 284">Comment</th> <th data-bbox="1854 201 1928 898"></th> </tr> </thead> <tbody> <tr> <td>START:</td> <td>IN</td> <td></td> <td>// INPUT character</td> <td rowspan="2">}</td> </tr> <tr> <td></td> <td>STO</td> <td>CHAR1</td> <td>// store in CHAR1</td> </tr> <tr> <td></td> <td>IN</td> <td></td> <td>// INPUT character</td> <td rowspan="2">}</td> </tr> <tr> <td></td> <td>STO</td> <td>CHAR2</td> <td>// store in CHAR2</td> </tr> <tr> <td></td> <td>LDD</td> <td>CHAR1</td> <td>// initialise ACC to ASCII value of CHAR1</td> <td>1</td> </tr> <tr> <td>LOOP:</td> <td>OUT</td> <td></td> <td>//output contents of ACC</td> <td>1+1</td> </tr> <tr> <td></td> <td>CMP</td> <td>CHAR2</td> <td>// compare ACC with CHAR2</td> <td>1</td> </tr> <tr> <td></td> <td>JPE</td> <td>ENDFOR</td> <td>// if equal jump to end of FOR loop</td> <td>1</td> </tr> <tr> <td></td> <td>INC</td> <td>ACC</td> <td>// increment ACC</td> <td>1</td> </tr> <tr> <td></td> <td>JMP</td> <td>LOOP</td> <td>// jump to LOOP</td> <td>1</td> </tr> <tr> <td>ENDFOR:</td> <td>END</td> <td></td> <td></td> <td></td> </tr> <tr> <td>CHAR1:</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>CHAR2:</td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	Label	Op code	Operand	Comment		START:	IN		// INPUT character	}		STO	CHAR1	// store in CHAR1		IN		// INPUT character	}		STO	CHAR2	// store in CHAR2		LDD	CHAR1	// initialise ACC to ASCII value of CHAR1	1	LOOP:	OUT		//output contents of ACC	1+1		CMP	CHAR2	// compare ACC with CHAR2	1		JPE	ENDFOR	// if equal jump to end of FOR loop	1		INC	ACC	// increment ACC	1		JMP	LOOP	// jump to LOOP	1	ENDFOR:	END				CHAR1:					CHAR2:					9
Label	Op code	Operand	Comment																																																																			
START:	IN		// INPUT character	}																																																																		
	STO	CHAR1	// store in CHAR1																																																																			
	IN		// INPUT character	}																																																																		
	STO	CHAR2	// store in CHAR2																																																																			
	LDD	CHAR1	// initialise ACC to ASCII value of CHAR1	1																																																																		
LOOP:	OUT		//output contents of ACC	1+1																																																																		
	CMP	CHAR2	// compare ACC with CHAR2	1																																																																		
	JPE	ENDFOR	// if equal jump to end of FOR loop	1																																																																		
	INC	ACC	// increment ACC	1																																																																		
	JMP	LOOP	// jump to LOOP	1																																																																		
ENDFOR:	END																																																																					
CHAR1:																																																																						
CHAR2:																																																																						
1(b)	<table border="1"> <thead> <tr> <th data-bbox="349 919 528 1002">Label</th> <th data-bbox="535 919 618 1002">Op code</th> <th data-bbox="624 919 792 1002">Operand</th> <th data-bbox="799 919 1554 1002">Comment</th> <th data-bbox="1854 919 1928 1382"></th> </tr> </thead> <tbody> <tr> <td>START:</td> <td>LDD</td> <td>NUMBER1</td> <td></td> <td>1</td> </tr> <tr> <td></td> <td>XOR</td> <td>MASK</td> <td>// convert to one's complement</td> <td>1</td> </tr> <tr> <td></td> <td>INC</td> <td>ACC</td> <td>// convert to two's complement</td> <td>1</td> </tr> <tr> <td></td> <td>STO</td> <td>NUMBER2</td> <td></td> <td>1</td> </tr> <tr> <td></td> <td>END</td> <td></td> <td></td> <td></td> </tr> <tr> <td>MASK:</td> <td>B11111111</td> <td></td> <td>// show value of mask in binary here</td> <td>1</td> </tr> <tr> <td>NUMBER1:</td> <td>B00000101</td> <td></td> <td>// positive integer</td> <td></td> </tr> <tr> <td>NUMBER2:</td> <td>B11111011</td> <td></td> <td>// show value of negative equivalent</td> <td>1</td> </tr> </tbody> </table>	Label	Op code	Operand	Comment		START:	LDD	NUMBER1		1		XOR	MASK	// convert to one's complement	1		INC	ACC	// convert to two's complement	1		STO	NUMBER2		1		END				MASK:	B11111111		// show value of mask in binary here	1	NUMBER1:	B00000101		// positive integer		NUMBER2:	B11111011		// show value of negative equivalent	1	6																							
Label	Op code	Operand	Comment																																																																			
START:	LDD	NUMBER1		1																																																																		
	XOR	MASK	// convert to one's complement	1																																																																		
	INC	ACC	// convert to two's complement	1																																																																		
	STO	NUMBER2		1																																																																		
	END																																																																					
MASK:	B11111111		// show value of mask in binary here	1																																																																		
NUMBER1:	B00000101		// positive integer																																																																			
NUMBER2:	B11111011		// show value of negative equivalent	1																																																																		

Question	Answer	Marks																																																							
2(a)	<ul style="list-style-type: none"> <li>A pointer that doesn't point to another node/other data/address // indicates the end of the branch</li> </ul>	<b>1</b>																																																							
2(b)	one mark per bullet <ul style="list-style-type: none"> <li>node with 'Athens' linked to left pointer of Berlin (ignore null pointer)</li> <li>null pointers in left and right pointers of Athens</li> </ul>	<b>2</b>																																																							
2(c)(i)	<table style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; border: none;">RootPointer</th> <th style="border: none;"></th> <th style="text-align: center; border: none;">LeftPointer</th> <th style="text-align: center; border: none;">Tree Data</th> <th style="text-align: center; border: none;">RightPointer</th> </tr> </thead> <tbody> <tr> <td style="border: 1px solid black; text-align: center; width: 100px;">0</td> <td style="border: none; padding: 0 10px;">[0]</td> <td style="border: 1px solid black; text-align: center;">2</td> <td style="border: 1px solid black; text-align: center;">Dublin</td> <td style="border: 1px solid black; text-align: center;">1</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none; padding: 0 10px;">[1]</td> <td style="border: 1px solid black; text-align: center;">-1/∅</td> <td style="border: 1px solid black; text-align: center;">London</td> <td style="border: 1px solid black; text-align: center;">3</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none; padding: 0 10px;">[2]</td> <td style="border: 1px solid black; text-align: center;">6</td> <td style="border: 1px solid black; text-align: center;">Berlin</td> <td style="border: 1px solid black; text-align: center;">5</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none; padding: 0 10px;">[3]</td> <td style="border: 1px solid black; text-align: center;">4</td> <td style="border: 1px solid black; text-align: center;">Paris</td> <td style="border: 1px solid black; text-align: center;">-1/∅</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none; padding: 0 10px;">[4]</td> <td style="border: 1px solid black; text-align: center;">-1/∅</td> <td style="border: 1px solid black; text-align: center;">Madrid</td> <td style="border: 1px solid black; text-align: center;">-1/∅</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none; padding: 0 10px;">[5]</td> <td style="border: 1px solid black; text-align: center;">-1/∅</td> <td style="border: 1px solid black; text-align: center;">Copenhagen</td> <td style="border: 1px solid black; text-align: center;">-1/∅</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none; padding: 0 10px;">[6]</td> <td style="border: 1px solid black; text-align: center;">-1/∅</td> <td style="border: 1px solid black; text-align: center;">Athens</td> <td style="border: 1px solid black; text-align: center;">-1/∅</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none; padding: 0 10px;">[7]</td> <td style="border: 1px solid black; text-align: center;">8</td> <td style="border: 1px solid black;"></td> <td style="border: 1px solid black; text-align: center;">-1/∅</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none; padding: 0 10px;">[8]</td> <td style="border: 1px solid black; text-align: center;">9</td> <td style="border: 1px solid black;"></td> <td style="border: 1px solid black; text-align: center;">-1/∅</td> </tr> <tr> <td style="border: none;"></td> <td style="border: none; padding: 0 10px;">[9]</td> <td style="border: 1px solid black; text-align: center;">-1/∅</td> <td style="border: 1px solid black;"></td> <td style="border: 1px solid black; text-align: center;">-1/∅</td> </tr> </tbody> </table> <p style="margin-top: 20px;">FreePointer</p> <p style="margin-left: 100px; border: 1px solid black; padding: 2px 10px; display: inline-block;">7</p> <p style="margin-left: 100px;"><b>1 mark</b></p>	RootPointer		LeftPointer	Tree Data	RightPointer	0	[0]	2	Dublin	1		[1]	-1/∅	London	3		[2]	6	Berlin	5		[3]	4	Paris	-1/∅		[4]	-1/∅	Madrid	-1/∅		[5]	-1/∅	Copenhagen	-1/∅		[6]	-1/∅	Athens	-1/∅		[7]	8		-1/∅		[8]	9		-1/∅		[9]	-1/∅		-1/∅	<b>5</b>
RootPointer		LeftPointer	Tree Data	RightPointer																																																					
0	[0]	2	Dublin	1																																																					
	[1]	-1/∅	London	3																																																					
	[2]	6	Berlin	5																																																					
	[3]	4	Paris	-1/∅																																																					
	[4]	-1/∅	Madrid	-1/∅																																																					
	[5]	-1/∅	Copenhagen	-1/∅																																																					
	[6]	-1/∅	Athens	-1/∅																																																					
	[7]	8		-1/∅																																																					
	[8]	9		-1/∅																																																					
	[9]	-1/∅		-1/∅																																																					
2(c)(ii)	<ul style="list-style-type: none"> <li>-1</li> <li>It is not the number for any node.</li> </ul>	<b>2</b>																																																							

Question	Answer	Marks
2(d)(i)	<pre> TYPE Node     LeftPointer : INTEGER     RightPointer : INTEGER     Data : STRING ENDTYPE  DECLARE Tree : ARRAY[0 : 9] OF Node  DECLARE FreePointer : INTEGER DECLARE RootPointer : INTEGER  PROCEDURE CreateTree()     DECLARE Index : INTEGER      RootPointer ← -1      FreePointer ← 0      FOR Index ← 0 TO 9 // link nodes         Tree[Index].LeftPointer ← Index + 1         Tree[Index].RightPointer ← -1     ENDFOR      Tree[9].LeftPointer ← -1 ENDPROCEDURE </pre>	<p style="text-align: right;"><b>7</b></p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p>

Question	Answer	Marks
2(d)(ii)	<pre> PROCEDURE AddToTree (ByVal NewDataItem : STRING) // if no free node report an error IF FreePointer = -1 THEN     ERROR("No free space left") ELSE // add new data item to first node in the free list     NewNodePointer ← FreePointer     <b>Tree [NewNodePointer] .Data ← NewDataItem</b>     // adjust free pointer     FreePointer ← <b>Tree [FreePointer] .LeftPointer</b>     // clear left pointer     Tree [NewNodePointer] .LeftPointer ← -1     // is tree currently empty ?     IF <b>RootPointer = -1</b>     THEN // make new node the root node         <b>RootPointer ← NewNodePointer</b>     ELSE // find position where new node is to be added         Index ← RootPointer         CALL FindInsertionPoint (NewDataItem, Index, Direction) </pre>	<p style="text-align: right;"><b>8</b></p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p>

Question	Answer	Marks
	<pre> IF Direction = "Left"     THEN // add new node on left <b>Tree[Index].LeftPointer ← NewNodePointer</b>     ELSE // add new node on right <b>Tree[Index].RightPointer ← NewNodePointer</b>     ENDIF ENDIF ENDIF ENDIF ENDPROCEDURE </pre>	<p>1</p> <p>1</p>
2(e)	<p><b>1 mark per bullet</b></p> <ul style="list-style-type: none"> <li>• test for base case (null/-1)</li> <li>• recursive call for left pointer</li> <li>• output data</li> <li>• recursive call for right pointer</li> <li>• order, visit left, output, visit right</li> </ul> <pre> IF Pointer &lt;&gt; NULL     THEN         TraverseTree(Tree[Pointer].LeftPointer)         OUTPUT Tree[Pointer].Data         TraverseTree(Tree[Pointer].RightPointer)     ENDIF ENDPROCEDURE </pre>	<p><b>5</b></p> <p>1</p> <p>1</p> <p>1 + 1</p> <p>1</p>

Question	Answer	Marks
3(a)	<p><b>1 mark per bullet</b></p> <ul style="list-style-type: none"> <li>• Instantiation of island object and calling DisplayGrid</li> <li>• Loop 3 times and Island.HideTreasure</li> <li>• Call procedures StartDig and DisplayGrid</li> </ul> <p><b>Example Python</b></p> <pre> Island = IslandClass() DisplayGrid() for Treasure in range(3):     Island.HideTreasure() StartDig() DisplayGrid() </pre> <p><b>Example Pascal</b></p> <pre> var Island : IslandClass; var Treasure : integer; begin     Island := IslandClass.Create();     DisplayGrid;     for Treasure := 1 to 3 do         Island.HideTreasure();     StartDig;     DisplayGrid; end; </pre>	<p style="text-align: center;"><b>3</b></p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p> <p style="text-align: right;">1</p>

Question	Answer	Marks
	<p><b>Example VB.NET</b></p> <pre> Dim Island As New IslandClass() DisplayGrid() For Treasure = 1 To 3     Island.HideTreasure() Next StartDig() DisplayGrid() </pre>	<p>1</p> <p>1</p> <p>1</p>

AlevelCSOnline.CF



Question	Answer	Marks
3(b)	<p><b>1 mark per bullet to max 5</b></p> <ul style="list-style-type: none"> <li>• Class heading and ending (in appropriate place)</li> <li>• Constructor heading and ending (in appropriate place)</li> <li>• Declaring grid with correct dimensions (as private)</li> <li>• Declaring Sand as a constant</li> <li>• Nested loops covering dimensions (0 – 29 and 0 – 9)</li> <li>• Assigning Sand // '.' to each array element</li> </ul> <p><b>Example Python</b></p> <pre>class IslandClass:     def __init__(self):         Sand = '.'         self.__Grid = [[Sand for j in range(30)]                        for i in range(10)]</pre> <p><b>Example Pascal</b></p> <pre>type IslandClass = class private     Grid : array[0..9, 0..29] of char; public     constructor Create();     procedure HideTreasure();     procedure DigHole(x, y : integer);     function GetSquare(x, y : integer) : char; end; constructor IslandClass.Create(); const Sand = '.'; var i, j : integer; begin     for i := 0 to 9 do         for j := 0 to 29 do             Grid[i, j] := Sand;         end; end;</pre>	<p style="text-align: right;"><b>5</b></p> <p style="text-align: right;">1 1 1 1 + 1 1  1 1  1 1  1 1</p>

Question	Answer	Marks
	<p><b>Example VB.NET</b></p> <pre> Class IslandClass   Private Grid (9, 29) As Char   Public Sub New()     Const Sand = "."     For i = 0 To 9       For j = 0 To 29         Grid(i, j) = Sand       Next     Next   End Sub End Class </pre>	<p>1 1 1 1 1 1</p>
3(c)(i)	<p><b>1 mark per bullet</b></p> <ul style="list-style-type: none"> <li>• Method (getter or property) heading, takes two parameters returns char, and ending</li> <li>• Method returns Grid value</li> </ul> <p><b>Example Python</b></p> <pre> def GetSquare(self, Row, Column) :     return self.__Grid[Row][Column] </pre> <p><b>Example Pascal</b></p> <pre> function IslandClass.GetSquare( Row, Column : integer) As Char; begin   Result := Grid[Row, Column]; end; </pre> <p><b>Example VB.NET</b></p> <pre> Public Function GetSquare(Row As Integer, Column As Integer) As Char   Return Grid(Row, Column) end Function </pre>	<p><b>2</b></p> <p>1 1 1 1 1</p>

Question	Answer	Marks
3(c)(ii)	<p>1 mark per bullet</p> <ul style="list-style-type: none"> <li>• DisplayGrid header and ending, with two loops with correct limits</li> <li>• Calling <b>Island</b>.GetSquare with correct parameters inside iteration</li> <li>• Output an entire row in one line</li> <li>• Output a new line at the end of a row</li> </ul> <p><b>Example Python</b></p> <pre>def DisplayGrid() :     for i in range (10) :         for j in range (30) :             print(island.GetSquare(i, j), end='')         print()</pre> <p><b>Example Pascal</b></p> <pre>procedure DisplayGrid(): var i, j : integer; begin     for i := 0 to 9 do         begin             for j := 0 to 29 do                 write(island.GetSquare(i, j));             writeln;         end;     end;</pre> <p><b>Example VB.NET</b></p> <pre>Sub DisplayGrid()     For i = 0 to 9         For j = 0 to 29             Console.Write(island.GetSquare(i, j))         Next         Console.WriteLine()     Next End Sub</pre>	<p style="text-align: right;"><b>4</b></p> <p style="text-align: right;">1 1 + 1 1</p> <p style="text-align: right;">1 1 + 1 1</p> <p style="text-align: right;">1 1 + 1 1</p>

Question	Answer	Marks
3(d)	<p><b>1 mark per bullet to max 5</b></p> <ul style="list-style-type: none"> <li>• Method header and Declaring Treasure as a constant</li> <li>• Generating a random number for column</li> <li>• Generating a random number for row</li> <li>• Check whether treasure already at <u>generated</u> location</li> <li>• Repeatedly generate new coordinates in a loop</li> <li>• Assign Treasure to location</li> </ul> <p><b>Example Python</b></p> <pre>def HideTreasure(self):     Treasure = 'T'     x = randint(0,9)     y = randint(0,29)     while self.__Grid[y][x] == Treasure:         x = randint(0,9)         y = randint(0,29)     self.__Grid[y][x] = Treasure</pre> <p><b>Example Pascal</b></p> <pre>procedure IslandClass.HideTreasure(); const Treasure = 'T'; var x, y : integer; begin     repeat         x := Random(10);         y := random(30);     until Grid[x, y] &lt;&gt; Treasure;     Grid[x, y] := Treasure; end;</pre>	<p><b>Max 5</b></p> <p>1</p> <p>1</p> <p>1</p> <p>1+1</p> <p>1</p> <p>1</p> <p>1</p> <p>1+1</p> <p>1</p>

Question	Answer	Marks
	<p><b>Example VB.NET</b></p> <pre> Public Sub HideTreasure()     Const Treasure = "T"     Dim RandomNumber As New Random     Dim x, y As Integer     Do         x = RandomNumber.Next(0, 10)         y = RandomNumber.Next(0, 30)         Loop Until Grid(x, y) &lt;&gt; Treasure         Grid(x, y) = Treasure     End Sub </pre>	<p>1</p> <p>1</p> <p>1</p> <p>1+1</p> <p>1</p>

A Level CS Online .CF

Question	Answer	Marks
3(e)(i)	<p><b>1 mark per bullet</b></p> <ul style="list-style-type: none"> <li>• Method heading, with two parameters &amp; Declaring constants for Treasure, Hole and FoundTreasure</li> <li>• Check if treasure at parameter locations</li> <li>• Set to FoundTreasure (X) and Set to Hole (O)</li> </ul> <p><b>Example Python</b></p> <pre> def DigHole(self, x, y) :     Treasure = 'T'     Hole = 'O'     Foundtreasure = 'X'     if self.__Grid[x][y] == Treasure:         self.__Grid[x][y] = Foundtreasure     else :         self.__Grid[x][y] = Hole     return </pre> <p><b>Example Pascal</b></p> <pre> procedure IslandClass.DigHole(x, y : integer); const Treasure = 'T'; const Hole = 'O'; const Foundtreasure = 'X'; begin     if Grid[x, y] = Treasure     then         Grid[x, y] := Foundtreasure     else         Grid[x, y] := Hole; end; </pre>	<p style="text-align: center;"><b>3</b></p> <p style="text-align: right;">1 1 1  1 1 1</p>

Question	Answer	Marks
	<p><b>Example VB.NET</b></p> <pre> Public Sub DigHole(x As Integer, y As Integer)   Const Treasure = "T"   Const Hole = "O"   Const Foundtreasure = "X"   If Grid(x, y) = Treasure Then     Grid(x, y) = Foundtreasure   Else     Grid(x, y) = Hole   End If End Sub </pre> <p>Diagrammatic marking: A right-facing curly bracket groups the three constant declarations (Treasure, Hole, Foundtreasure) and is aligned with a '1' in the Marks column. A second right-facing curly bracket groups the If-Else conditional block and is aligned with a '1' in the Marks column. A third right-facing curly bracket groups the End If statement and is aligned with a '1' in the Marks column.</p>	<p>1</p> <p>1</p> <p>1</p>

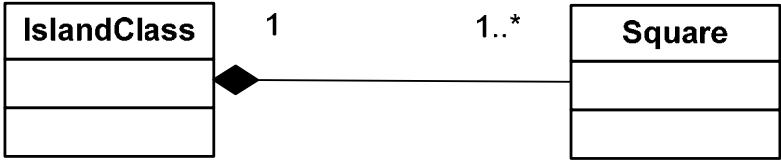
A Level CS Online .CF

Question	Answer	Marks
3(e)(ii)	<p><b>1 mark per bullet to max 5</b></p> <ul style="list-style-type: none"> <li>• Prompt to user for position down and across, read positions input as an IntegerValidation for position row – between 0 and 9</li> <li>• Validation for position column- between 0 and 29</li> <li>• Exception handling/pass for validation</li> <li>• Ask for repeated input until valid (for both row and column)</li> <li>• Call Island.DigHole method with the coordinates</li> </ul> <p><b>Example Python</b></p> <pre> def StartDig() :     Valid = False     while not Valid :          # validate down position         try:             x = int(input("position down &lt;0 to 9&gt; ? "))             if x &gt;= 0 and x &lt;= 9 :                 Valid = True         except:             Valid = False     Valid = False     while not Valid :          # validate across position         try :             y = int(input("position across &lt;0 to 29&gt; ? "))             if y &gt;= 0 and y &lt;= 29 :                 Valid = True         except :             Valid = False     island.DigHole(x, y)     return </pre>	<p><b>Max 5</b></p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p>



Question	Answer	Marks
	<p><b>Example Pascal</b></p> <pre> procedure StartDig; var xString, yString : String;     x, y : integer; begin     Valid := False;     repeat         Write('position down &lt;0 to 9&gt;? '); ReadLn(xString);         try             x := StrToInt(xString);             if (x &gt;= 0) AND (x &lt;= 9)             then                 Valid := True;         except             Valid := False;     until Valid;     Valid := False;     repeat         Write(position across &lt;0 to 29&gt; ? '); ReadLn(yString);         try             y := StrToInt(yString);             if (y &gt;= 0) AND (y &lt;= 29)             then                 Valid := True;         except             Valid := False;     until Valid;     island.DigHole(x,y); end; </pre>	<p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p>

Question	Answer	Marks
	<p><b>Example VB.NET</b></p> <pre> Sub StartDig()   Dim x, y As Integer   Dim Valid = False   Do     Console.WriteLine("Position down &lt;0 to 9&gt;? ")     Try       x = CInt(Console.ReadLine())       If (x &gt;= 0) AND (x &lt;= 9) Then         Valid = True       End If     Catch       Valid = False 'accept different types of exceptions     End Try   Loop Until Valid Valid = False Do   Console.WriteLine("Position across &lt;0 to 29&gt; ? ")   Try     y = int(Console.ReadLine())     If (y &gt;= 0) AND (y &lt;= 29) Then       Valid = True     End IF   Catch     Valid = False   End Try   Loop until Valid   island.DigHole(x, y) End Sub </pre> <p>Brackets in the original image group the first Try-If-End If block and the second Try-If-End IF block, each with a mark of 1. The Loop Until Valid and Loop until Valid lines each have a mark of 1.</p>	<p>1</p> <p>1</p> <p>1</p> <p>1</p>
3(f)(i)	containment/aggregation	1

Question	Answer	Marks
3(f)(ii)	<ul style="list-style-type: none"><li>• IslandClass box and Square Box, with correct connection</li><li>• One at IslandClass and one .. * at Square</li></ul>  <pre>classDiagram     class IslandClass     class Square     IslandClass "1" *-- "1..*" Square</pre>	<b>Max 2</b>

AlevelCSOnline.CF