

---

**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.

Question	Answer				Marks
1(a)	<b>Label</b>	<b>Op code</b>	<b>Operand</b>	<b>Comment</b>	<div> <div> } 1 </div> <div> } 1 </div> 1 1+1 1 1 1 1 1 </div>
	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	
	LOOP:	OUT		//output contents of ACC	
		CMP	CHAR2	// compare ACC with CHAR2	
		JPE	ENDFOR	// if equal jump to end of FOR loop	
		INC	ACC	// increment ACC	
		JMP	LOOP	// jump to LOOP	
	ENDFOR:	END			
	CHAR1:				
	CHAR2:				
1(b)	<b>Label</b>	<b>Op code</b>	<b>Operand</b>	<b>Comment</b>	<div> 1 1 1 1 1 1 1 </div>
	START:	LDD	NUMBER1		
		XOR	MASK	// convert to one's complement	
		INC	ACC	// convert to two's complement	
		STO	NUMBER2		
		END			
	MASK:	B11111111		// show value of mask in binary here	
	NUMBER1:	B00000101		// positive integer	
	NUMBER2:	B11111011		// show value of negative equivalent	

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)	<div style="display: flex; align-items: center; justify-content: center;"> <div style="text-align: center; margin-right: 20px;"> <p>RootPointer</p> <div style="border: 1px solid black; padding: 5px; width: 60px; margin: 0 auto;">0</div> </div> <div style="margin-right: 20px;">[0]</div> <table border="1" style="border-collapse: collapse; text-align: center;"> <thead> <tr> <th>LeftPointer</th><th>Tree Data</th><th>RightPointer</th></tr> </thead> <tbody> <tr><td>2</td><td>Dublin</td><td>1</td></tr> <tr><td>-1/Ø</td><td>London</td><td>3</td></tr> <tr><td>6</td><td>Berlin</td><td>5</td></tr> <tr><td>4</td><td>Paris</td><td>-1/Ø</td></tr> <tr><td>-1/Ø</td><td>Madrid</td><td>-1/Ø</td></tr> <tr><td>-1/Ø</td><td>Copenhagen</td><td>-1/Ø</td></tr> <tr><td>-1/Ø</td><td>Athens</td><td>-1/Ø</td></tr> <tr><td>8</td><td></td><td>-1/Ø</td></tr> <tr><td>9</td><td></td><td>-1/Ø</td></tr> <tr><td>-1/Ø</td><td></td><td>-1/Ø</td></tr> </tbody> </table> </div> <div style="margin-top: 20px; margin-left: 100px;"> <p>FreePointer</p> <div style="border: 1px solid black; padding: 5px; width: 60px; margin: 0 auto;">7</div> <p><b>1 mark</b></p> </div>	LeftPointer	Tree Data	RightPointer	2	Dublin	1	-1/Ø	London	3	6	Berlin	5	4	Paris	-1/Ø	-1/Ø	Madrid	-1/Ø	-1/Ø	Copenhagen	-1/Ø	-1/Ø	Athens	-1/Ø	8		-1/Ø	9		-1/Ø	-1/Ø		-1/Ø	<b>5</b>
LeftPointer	Tree Data	RightPointer																																	
2	Dublin	1																																	
-1/Ø	London	3																																	
6	Berlin	5																																	
4	Paris	-1/Ø																																	
-1/Ø	Madrid	-1/Ø																																	
-1/Ø	Copenhagen	-1/Ø																																	
-1/Ø	Athens	-1/Ø																																	
8		-1/Ø																																	
9		-1/Ø																																	
-1/Ø		-1/Ø																																	
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>7</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>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><b>8</b></p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>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  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>

**PUBLISHED**

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><b>3</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 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>



**PUBLISHED**

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><b>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</p> <p>1</p> <p>1</p> <p>1</p> <p>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 style="text-align: right;">} </p>	<p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>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>2</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>1</p> <p>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><b>4</b></p> <p>1 1 + 1 1</p> <p>1 1 + 1 1</p> <p>1 1 + 1 1</p>

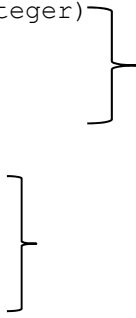
**PUBLISHED**

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>



**PUBLISHED**


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><b>3</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 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>1</p> <p>1</p> <p>1</p>

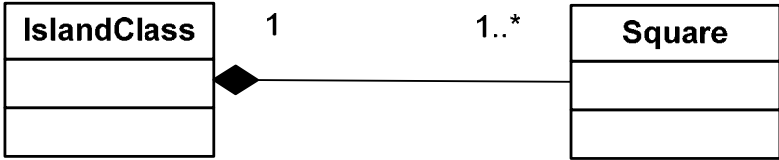
**PUBLISHED**

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>Diagrammatic marking for the code above:</p> <ul style="list-style-type: none"> <li>Try block (lines 10-14): 1 mark</li> <li>If statement (line 12): 1 mark</li> <li>Try block (lines 24-28): 1 mark</li> <li>If statement (line 26): 1 mark</li> <li>Loop until Valid (line 20): 1 mark</li> <li>island.DigHole(x, y) (line 22): 1 mark</li> </ul>	<p>1</p> <p>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>