Cambridge International AS & A Level

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®, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

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

CAMBRIDGE
International Examinations

Question	Answer					Marks
1(a)						**9**

Label	Op code	Operand	Comment	
START:	**IN**		// INPUT character	⎫ 1
	STO	**CHAR1**	// store in CHAR1	⎬
	IN		// INPUT character	⎫ 1
	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:				

Question	Answer				Marks
1(b)					**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)	• A pointer that doesn't point to another node/other data/address // indicates the end of the branch	1
2(b)	one mark per bullet • node with 'Athens' linked to left pointer of Berlin (ignore null pointer) • null pointers in left and right pointers of Athens	2
2(c)(i)		5

RootPointer

	0

FreePointer

	7

1 mark

	LeftPointer	Tree Data	RightPointer
[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/∅

Question	Answer	Marks
2(c)(ii)	• −1 • It is not the number for any node.	2

Question	Answer	Marks

2(d)(i)	```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```
		7
		
		1
		
		1
		
		1
		
		1
		
		1
		
		1
		
		1

Question	Answer	Marks
2(d)(ii)	```PROCEDURE AddToTree(ByVal NewDataItem : STRING)``` ```// if no free node report an error```	**8**

```
2(d)(ii)    PROCEDURE AddToTree(ByVal NewDataItem : STRING)

            // if no free node report an error

                IF FreePointer = -1                                              1

                    THEN

                        ERROR("No free space left")

                    ELSE    // add new data item to first node in the free list

                        NewNodePointer ← FreePointer

                        Tree[NewNodePointer].Data ← NewDataItem                  1

                        // adjust free pointer

                        FreePointer ← Tree[FreePointer].LeftPointer              1

                        // clear left pointer

                        Tree[NewNodePointer].LeftPointer ← -1                    1

                        // is tree currently empty ?

                        IF RootPointer = -1                                      1

                            THEN    // make new node the root node

                                RootPointer ← NewNodePointer                     1

                            ELSE    // find position where new node is to be added

                                Index ← RootPointer

                                CALL FindInsertionPoint(NewDataItem, Index, Direction)
```

Question	Answer	Marks
| | ```
                IF Direction = "Left"
                    THEN   // add new node on left
            Tree[Index].LeftPointer ← NewNodePointer          1
                    ELSE   // add new node on right
            Tree[Index].RightPointer ← NewNodePointer          1
                    ENDIF
                ENDIF
            ENDIF
        ENDPROCEDURE
``` | |
| 2(e) | **1 mark per bullet**<br>• test for base case (null/-1)<br>• recursive call for left pointer<br>• output data<br>• recursive call for right pointer<br>• order, visit left, output, visit right<br><br>```
        IF Pointer <> NULL                                      1

            THEN

                TraverseTree(Tree[Pointer].LeftPointer)         1

                OUTPUT Tree[Pointer].Data                     1 + 1

                TraverseTree(Tree[Pointer].RightPointer)        1

            ENDIF

        ENDPROCEDURE
``` | 5 |

Question	Answer	Marks

3(a)	**1 mark per bullet**
	• Instantiation of island object and calling DisplayGrid
	• Loop 3 times and Island.HideTreasure
	• Call procedures StartDig and DisplayGrid
	
	Example Python
	
	`Island = IslandClass()` 1
	
	`DisplayGrid()`
	
	`for Treasure in range(3):`
	
	    `Island.HideTreasure()` 1
	
	`StartDig()`
	
	`DisplayGrid()` 1
	
	Example Pascal
	
	`var Island : IslandClass;`
	
	`var Treasure : integer;`
	
	`begin`
	
	    `Island := IslandClass.Create();` 1
	
	    `DisplayGrid;`
	
	    `for Treasure := 1 to 3 do`
	
	      `Island.HideTreasure();` 1
	
	    `StartDig;`
	
	    `DisplayGrid;` 1
	
	`end;`
		3

Question	Answer	Marks
	Example VB.NET	
| | ```
Dim Island As New IslandClass()
DisplayGrid()
For Treasure = 1 To 3
    Island.HideTreasure()
Next
StartDig()
DisplayGrid()
``` | 1<br><br>1<br><br><br>1 |

Question	Answer	Marks

3(b)	**1 mark per bullet to max 5**
	• Class heading and ending (in appropriate place)
	• Constructor heading and ending (in appropriate place)
	• Declaring grid with correct dimensions (as private)
	• Declaring Sand as a constant
	• Nested loops covering dimensions (0 – 29 and 0 – 9)
	• Assigning Sand // '.' to each array element
	
	Example Python
	<pre>class IslandClass: 1
	   def __init__(self): 1
	      Sand = '.' 1
	      self.__Grid = [[Sand for j in range(30)] 1 + 1
	                            for i in range(10)] 1</pre>
	Example Pascal
	<pre>type
	IslandClass = class 1
	private
	   Grid : array[0..9, 0..29] of char; 1
	public
	   constructor Create();
	   procedure HideTreasure();
	   procedure DigHole(x, y : integer);
	   function GetSquare(x, y : integer) : char;
	end;
	constructor IslandClass.Create(); 1
	   const Sand = '.'; 1
	   var i, j : integer;
	   begin
	      for i := 0 to 9 do
	         for j := 0 to 29 do 1
	            Grid[i, j] := Sand; 1
	   end;</pre>
		5

Question	Answer	Marks
| | **Example VB.NET**<br>```
Class IslandClass                                          1
   Private Grid (9, 29) As Char                            1
   Public Sub New()                                        1
      Const Sand = "."                                     1
      For i = 0 To 9
         For j = 0 To 29                                   1
            Grid(i, j) = Sand                              1
         Next
      Next
   End Sub
End Class
``` | |
| 3(c)(i) | **1 mark per bullet**<br><br>•   Method (getter or property) heading, takes two parameters returns char, and ending<br>•   Method returns Grid value<br><br>**Example Python**<br>```
def GetSquare(self, Row, Column) :                         1
   return self.__Grid[Row][Column]                         1
```<br>**Example Pascal**<br>```
function IslandClass.GetSquare( Row, Column : integer) As Char;   1
begin
   Result := Grid[Row, Column];
end;                                                       1
```<br>**Example VB.NET**<br>```
Public Function GetSquare(Row As Integer, Column As Integer) As Char   1
   Return Grid(Row, Column)                                1
end Function
``` | **2** |

Question	Answer	Marks

3(c)(ii)	1 mark per bullet
	• DisplayGrid header and ending, with two loops with correct limits
	• Calling **Island.**GetSquare with correct parameters inside iteration
	• Output an entire row in one line
	• Output a new line at the end of a row
	
	Example Python
	<pre>def DisplayGrid() :
	   for i in range (10) :
	      for j in range (30) :
	         print(island.GetSquare(i, j), end='')
	      print()</pre>
	Example Pascal
	<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>
	Example VB.NET
	<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>
		4
		
		1
		1 + 1
		1
		
		1
		1 + 1
		1
		
		1
		1 + 1
		
		1

Question	Answer	Marks

3(d)	**1 mark per bullet to max 5**
	• Method header and Declaring Treasure as a constant
	• Generating a random number for column
	• Generating a random number for row
	• Check whether treasure already at <u>generated</u> location
	• Repeatedly generate new coordinates in a loop
	• Assign Treasure to location
	
	Example Python
	<pre>def HideTreasure(self): 1
	   Treasure = 'T'
	   x = randint(0,9) 1
	   y = randint(0,29) 1
	   while self.__Grid[y][x] == Treasure: 1+1
	      x = randint(0,9)
	      y = randint(0,29)
	   self.__Grid[y][x] = Treasure 1</pre>
	Example Pascal
	<pre>procedure IslandClass.HideTreasure();
	   const Treasure = 'T'; 1
	   var x, y : integer;
	   begin
	      repeat
	         x := Random(10); 1
	         y := random(30); 1
	      until Grid[x, y] <> Treasure; 1+1
	      Grid[x, y] := Treasure; 1
	   end;</pre>
		Max 5

Question	Answer	Marks
	Example VB.NET	

```
Public Sub HideTreasure()
    Const Treasure = "T"                              1
    Dim RandomNumber As New Random
    Dim x, y As Integer
    Do
        x = RandomNumber.Next(0, 10)                  1
        y = RandomNumber.Next(0, 30)                  1
    Loop Until Grid(x, y) <> Treasure              1+1
    Grid(x, y) = Treasure                             1
End Sub
```

Question	Answer	Marks

3(e)(i)	**1 mark per bullet**
	• Method heading, with two parameters & Declaring constants for Treasure, Hole and FoundTreasure
	• Check if treasure at parameter locations
	• Set to FoundTreasure (X) and Set to Hole (O)
	
	Example Python
	<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> 1
	
	    1
	
	    1
	
	Example Pascal
	<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> 1
	
	    1
	
	    1
		3

Question	Answer	Marks

	Example VB.NET
	``
	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
	``
		1
		
		1
		
		1

Question	Answer	Marks
3(e)(ii)	**1 mark per bullet to max 5** • Prompt to user for position down and across, read positions input as an IntegerValidation for position row – between 0 and 9 • Validation for position column- between 0 and 29 • Exception handling/pass for validation • Ask for repeated input until valid (for both row and column) • Call Island.DigHole method with the coordinates	**Max 5**

Example Python

```
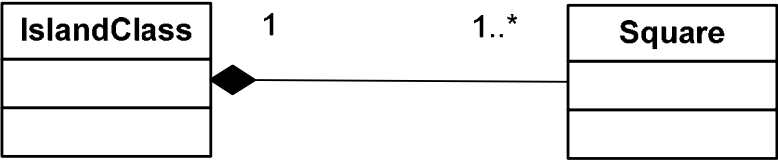def StartDig() :
    Valid = False
    while not Valid :       # validate down position               1
        try:
            x = int(input("position down <0 to 9> ? "))            1
            if x >= 0 and x <= 9 :                                 1
                Valid = True
        except:
            Valid = False
    Valid = False
    while not Valid :       # validate across position
        try :
            y = int(input("position across <0 to 29> ? "))        1
            if y >= 0 and y <= 29 :                               1
                Valid = True
        except :
            Valid = False
    island.DigHole(x, y)                                          1
    return
```

Question	Answer	Marks

	Example Pascal
	
	``pascal
	procedure StartDig;
	var xString, yString : String;
	   x, y : integer;
	begin
	   Valid := False;
	   repeat
	      Write('position down <0 to 9>? '); ReadLn(xString);
	      try
	         x := StrToInt(xString);
	         if (x >= 0) AND (x <= 9)
	            then
	               Valid := True;
	      except
	         Valid := False;
	   until Valid;
	   Valid := False;
	   repeat
	      Write(position across <0 to 29> ? '); ReadLn(yString);
	      try
	         y := StrToInt(yString);
	         if (y >= 0) AND (y <= 29)
	            then
	               Valid := True;
	      except
	         Valid := False;
	   until Valid;
	   island.DigHole(x,y);
	end;
	``
		1
		
		1
		
		1
		
		1
		
		1
		1

Question	Answer	Marks

	Example VB.NET
	```Sub StartDig()```
	``` Dim x, y As Integer```
	``` Dim Valid = False```
	``` Do```
	``` Console.Write("Position down <0 to 9>? ")```
	``` Try```
	``` x = CInt(Console.ReadLine())```
	``` If (x >= 0) AND (x <= 9) Then```
	``` Valid = True```
	``` End If```
	``` Catch```
	``` Valid = False 'accept different types of exceptions```
	``` End Try```
	``` Loop Until Valid```
	```Valid = False```
	``` Do```
	``` Console.Write("Position across <0 to 29> ? ")```
	``` Try```
	``` y = int(Console.ReadLine())```
	``` If (y >= 0) AND (y <= 29) Then```
	``` Valid = True```
	``` End IF```
	``` Catch```
	``` Valid = False```
	``` End Try```
	``` Loop until Valid```
	``` island.DigHole(x, y)```
	```End Sub```
		1
		1
		
		1
		1
		
		1
		1

3(f)(i)	containment/aggregation	**1**

Question	Answer	Marks
3(f)(ii)	• IslandClass box and Square Box, with correct connection • One at IslandClass and one .. * at Square IslandClass 1 1..* Square	**Max 2**