
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.

Question	Answer				Marks
1(a)	Label	Op code	Operand	Comment	} 1 } 1 1 1+1 1 1 1 1 1
	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)	Label	Op code	Operand	Comment	1 1 1 1 1 1 1
	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)	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)	RootPointer 0 [0] [1] [2] [3] [4] [5] [6] [7] [8] [9] <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/∅</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/∅</td></tr> <tr><td>-1/∅</td><td>Madrid</td><td>-1/∅</td></tr> <tr><td>-1/∅</td><td>Copenhagen</td><td>-1/∅</td></tr> <tr><td>-1/∅</td><td>Athens</td><td>-1/∅</td></tr> <tr><td>8</td><td></td><td>-1/∅</td></tr> <tr><td>9</td><td></td><td>-1/∅</td></tr> <tr><td>-1/∅</td><td></td><td>-1/∅</td></tr> </tbody> </table> FreePointer 7 1 mark	LeftPointer	Tree Data	RightPointer	2	Dublin	1	-1/∅	London	3	6	Berlin	5	4	Paris	-1/∅	-1/∅	Madrid	-1/∅	-1/∅	Copenhagen	-1/∅	-1/∅	Athens	-1/∅	8		-1/∅	9		-1/∅	-1/∅		-1/∅	5
LeftPointer	Tree Data	RightPointer																																	
2	Dublin	1																																	
-1/∅	London	3																																	
6	Berlin	5																																	
4	Paris	-1/∅																																	
-1/∅	Madrid	-1/∅																																	
-1/∅	Copenhagen	-1/∅																																	
-1/∅	Athens	-1/∅																																	
8		-1/∅																																	
9		-1/∅																																	
-1/∅		-1/∅																																	
2(c)(ii)	-1 - It is not the number for any node.	2																																	

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>	7 1 1 1 1 1 1

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 Tree[NewNodePointer].Data ← NewDataItem // adjust free pointer FreePointer ← Tree[FreePointer].LeftPointer // clear left pointer Tree[NewNodePointer].LeftPointer ← -1 // is tree currently empty ? IF RootPointer = -1 THEN // make new node the root node RootPointer ← NewNodePointer ELSE // find position where new node is to be added Index ← RootPointer CALL FindInsertionPoint(NewDataItem, Index, Direction) </pre>	8 1 1 1 1 1 1

Question	Answer	Marks
	<pre> IF Direction = "Left" THEN // add new node on left Tree[Index].LeftPointer ← NewNodePointer ELSE // add new node on right Tree[Index].RightPointer ← NewNodePointer ENDIF ENDIF ENDIF ENDPROCEDURE </pre>	1 1
2(e)	1 mark per bullet • test for base case (null/-1) • recursive call for left pointer • output data • recursive call for right pointer • order, visit left, output, visit right <pre> IF Pointer <> NULL THEN TraverseTree(Tree[Pointer].LeftPointer) OUTPUT Tree[Pointer].Data TraverseTree(Tree[Pointer].RightPointer) ENDIF ENDPROCEDURE </pre>	5 1 1 1 + 1 1

PUBLISHED

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 <pre> Island = IslandClass() DisplayGrid() for Treasure in range(3): Island.HideTreasure() StartDig() DisplayGrid() </pre> Example Pascal <pre> var Island : IslandClass; var Treasure : integer; begin Island := IslandClass.Create(); DisplayGrid; for Treasure := 1 to 3 do Island.HideTreasure(); StartDig; DisplayGrid; end; </pre>	3 1 1 1 1 1 1

Question	Answer	Marks
	Example VB.NET <pre> Dim Island As New IslandClass() DisplayGrid() For Treasure = 1 To 3 Island.HideTreasure() Next StartDig() DisplayGrid() </pre>	1 1 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: def __init__(self): Sand = '.' self.__Grid = [[Sand for j in range(30)] for i in range(10)]</pre> Example Pascal <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>	5 1 1 1 1 + 1 1 1 1 1 1 1 1 1

Question	Answer	Marks
	Example VB.NET <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> 1 1 1 1 1 1	
3(c)(i)	1 mark per bullet • Method (getter or property) heading, takes two parameters returns char, and ending • Method returns Grid value Example Python <pre> def GetSquare(self, Row, Column) : return self.__Grid[Row][Column] </pre> 1 1 Example Pascal <pre> function IslandClass.GetSquare(Row, Column : integer) As Char; begin Result := Grid[Row, Column]; end; </pre> 1 1 Example VB.NET <pre> Public Function GetSquare(Row As Integer, Column As Integer) As Char Return Grid(Row, Column) end Function </pre> 1 1	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

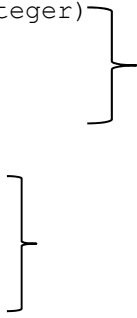
PUBLISHED

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): 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> Example Pascal <pre>procedure IslandClass.HideTreasure(); const Treasure = 'T'; var x, y : integer; begin repeat x := Random(10); y := random(30); until Grid[x, y] <> Treasure; Grid[x, y] := Treasure; end;</pre>	Max 5 1 1 1 1+1 1 1 1 1 1+1 1

Question	Answer	Marks
	Example VB.NET <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) <> Treasure Grid(x, y) = Treasure End Sub </pre>	1 1 1 1+1 1

PUBLISHED

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> 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>	3 1 1 1 1 1 1

Question	Answer	Marks
	Example VB.NET <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> 	1 1 1

PUBLISHED

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 Example Python <pre>def StartDig() : Valid = False while not Valid : # validate down position try: x = int(input("position down <0 to 9> ? ")) if x >= 0 and x <= 9 : Valid = True except: Valid = False Valid = False while not Valid : # validate across position try : y = int(input("position across <0 to 29> ? ")) if y >= 0 and y <= 29 : Valid = True except : Valid = False island.DigHole(x, y) return</pre>	Max 5 1 1 1 1 1 1

Question	Answer	Marks
	Example Pascal <pre> 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; </pre> 	1 1 1 1 1

Question	Answer	Marks
	Example VB.NET <pre> Sub StartDig() Dim x, y As Integer Dim Valid = False Do Console.WriteLine("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.WriteLine("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 </pre>	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 <pre>classDiagram class IslandClass class Square IslandClass "1" *-- "1..*" Square</pre>	Max 2