



eduresourceza.com

Cambridge International AS & A Level

COMPUTER SCIENCE

9608/42

Paper 4 Further Problem-solving and Programming Skills

October/November 2021

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 International will not enter into discussions about these mark schemes.

Cambridge International is publishing the mark schemes for the October/November 2021 series for most Cambridge IGCSE™, Cambridge International A and AS Level components and some Cambridge O Level components.

This document consists of **21** printed pages.

PUBLISHED**Generic Marking Principles**

These general marking principles must be applied by all examiners when marking candidate answers. They should be applied alongside the specific content of the mark scheme or generic level descriptors for a question. Each question paper and mark scheme will also comply with these marking principles.

GENERIC MARKING PRINCIPLE 1:

Marks must be awarded in line with:

- the specific content of the mark scheme or the generic level descriptors for the question
- the specific skills defined in the mark scheme or in the generic level descriptors for the question
- the standard of response required by a candidate as exemplified by the standardisation scripts.

GENERIC MARKING PRINCIPLE 2:

Marks awarded are always **whole marks** (not half marks, or other fractions).

GENERIC MARKING PRINCIPLE 3:

Marks must be awarded **positively**:

- marks are awarded for correct/valid answers, as defined in the mark scheme. However, credit is given for valid answers which go beyond the scope of the syllabus and mark scheme, referring to your Team Leader as appropriate
- marks are awarded when candidates clearly demonstrate what they know and can do
- marks are not deducted for errors
- marks are not deducted for omissions
- answers should only be judged on the quality of spelling, punctuation and grammar when these features are specifically assessed by the question as indicated by the mark scheme. The meaning, however, should be unambiguous.

GENERIC MARKING PRINCIPLE 4:

Rules must be applied consistently, e.g. in situations where candidates have not followed instructions or in the application of generic level descriptors.

GENERIC MARKING PRINCIPLE 5:

Marks should be awarded using the full range of marks defined in the mark scheme for the question (however; the use of the full mark range may be limited according to the quality of the candidate responses seen).

GENERIC MARKING PRINCIPLE 6:

Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

Question	Answer	Marks
1(a)	1 mark per bullet point to max 4 • Set the first element to be the sorted list • Store the next element in a temporary variable // store the value to be sorted in a temporary variable • ... compare this next element to each element in the sorted list • Move the elements that are greater than it one space to the right and insert the temporary variable // swap the element down until in the correct positions • Loop through all items from 2nd to end of array/100	4
1(b)	1 mark for each completed statement <pre> PROCEDURE Bubble(ByRef NumberArray : ARRAY[0:99] OF INTEGER) DECLARE Outer : INTEGER DECLARE Swap : BOOLEAN DECLARE Inner : INTEGER DECLARE Temp : INTEGER Outer ← LENGTH(NumberArray) - 1 REPEAT Inner ← 0 Swap ← FALSE REPEAT IF NumberArray[Inner] > NumberArray[Inner + 1] THEN Temp ← NumberArray[Inner] NumberArray[Inner] ← NumberArray[Inner + 1] NumberArray[Inner + 1] ← Temp Swap ← TRUE ENDIF Inner ← Inner + 1 UNTIL Inner = Outer Outer ← Outer - 1 UNTIL Swap = FALSE OR Outer = 0 ENDPROCEDURE </pre>	5

Question	Answer	Marks
2	1 mark per bullet point: • input of Number1 Number2 Command on same level under an input box • three functions (1, 2, 3) below e.g. decision box ... • ...with selection on each • output value at end in box below calculate <pre>graph TD; calculate[calculate] --- H1[]; H1 --- INPUT[INPUT]; H1 --- Decision[Decision]; H1 --- OUTPUT[OUTPUT Value]; INPUT --- H2[]; H2 --- Number1[Number1]; H2 --- Number2[Number2]; H2 --- Command[Command]; Decision --- H3[]; H3 --- Function1["Function1 (Number1, Number2)"]; H3 --- Function2["Function2 (Number1, Number2)"]; H3 --- Function3["Function3 (Number1, Number2)"];</pre>	4

Question	Answer										Marks
3	1 mark for each row										3
			Rules								
Conditions	One or more upper-case letters		N	Y	N	Y	N	Y	N	Y	
	One or more numeric characters		N	N	Y	Y	N	N	Y	Y	
	One or more symbols		N	N	N	N	Y	Y	Y	Y	
Actions	Strong					Y		Y	Y	Y	
	Medium			Y	Y		Y				
	Weak		Y								

Question	Answer										Marks
4(a)	1 mark per bullet point - Record declaration with identifier Node all three fields declared with type integer Example: TYPE Node DECLARE LeftPointer : INTEGER DECLARE Data : INTEGER DECLARE RightPointer : INTEGER ENDTYPE										2

Question	Answer	Marks
4(b)	1 mark per bullet point: • Declaration with correct identifier (Node100) of type Node • Assigning LeftPointer to 1 and RightPointer to 4 • Assigning 100 to the Data Example pseudocode <pre> DECLARE Node100 : Node Node100.LeftPointer ← 1 Node100.Data ← 100 Node100.RightPointer ← 4 </pre>	3
4(c)(i)	To point to the start/first of the empty node/nodes	1
4(c)(ii)	–1 or below // 101 or above	1

Question	Answer	Marks
4(c)(iii)	1 mark for 23 at top, with 5 below left, 100 below right 1 mark for remaining in correct places below 5 and 100 <pre>graph TD; 23[23] --- 5[5]; 23 --- 100[100]; 5 --- B1[]; 5 --- 8[8]; 100 --- 88[88]; 100 --- B2[]; B1 --- B1_1[]; B1 --- B1_2[]; 8 --- B3[]; 8 --- 9[9]; 88 --- B4[]; 88 --- B4_1[]; B2 --- B5[]; B2 --- B5_1[]</pre>	2

Question	Answer	Marks
4(c)(iv)	1 mark for each completed statement <pre> PROCEDURE AddData(NewNode) BinaryTree[FreePointer] ← NewNode BinaryTree[FreePointer].LeftPointer ← -1 BinaryTree[FreePointer].RightPointer ← -1 DECLARE PositionFound : BOOLEAN DECLARE PointerCounter : INTEGER PositionFound ← FALSE PointerCounter ← RootNode WHILE NOT PositionFound IF NewNode.Data < BinaryTree[PointerCounter].Data THEN IF BinaryTree[PointerCounter].LeftPointer = -1 THEN BinaryTree[PointerCounter].LeftPointer ← FreePointer PositionFound ← TRUE ELSE PointerCounter ← BinaryTree[PointerCounter].LeftPointer ENDIF ELSE IF BinaryTree[PointerCounter].RightPointer = -1 THEN BinaryTree[PointerCounter].RightPointer ← FreePointer PositionFound ← True ELSE PointerCounter ← BinaryTree[PointerCounter].RightPointer ENDIF ENDIF ENDWHILE FreePointer ← FreePointer + 1 ENDPROCEDURE </pre>	5

Question	Answer	Marks																								
5(a)	1 mark per bullet point - Return value of 25 (in space or if space left empty look at tracing) - Calling with 1 and 15, then 2 and 15 - Calling with 4, then 8, then 16 - Showing the unwinding of the return values <table><thead><tr><th>Function Call</th><th>Num1</th><th>Num2</th><th>Return value</th></tr></thead><tbody><tr><td>Recursive(1, 15)</td><td>1</td><td>15</td><td>1 + Recursive(2, 15) 1 + 24 = 25</td></tr><tr><td>Recursive(2, 15)</td><td>2</td><td>15</td><td>2 + Recursive(4, 15) 2 + 22 = 24</td></tr><tr><td>Recursive(4, 15)</td><td>4</td><td>15</td><td>4 + Recursive(8, 15) 4 + 18 = 22</td></tr><tr><td>Recursive(8, 15)</td><td>8</td><td>15</td><td>8 + Recursive(16, 15) 8 + 10 = 18</td></tr><tr><td>Recursive(16, 15)</td><td>16</td><td>15</td><td>10</td></tr></tbody></table>	Function Call	Num1	Num2	Return value	Recursive(1, 15)	1	15	1 + Recursive(2, 15) 1 + 24 = 25	Recursive(2, 15)	2	15	2 + Recursive(4, 15) 2 + 22 = 24	Recursive(4, 15)	4	15	4 + Recursive(8, 15) 4 + 18 = 22	Recursive(8, 15)	8	15	8 + Recursive(16, 15) 8 + 10 = 18	Recursive(16, 15)	16	15	10	4
Function Call	Num1	Num2	Return value																							
Recursive(1, 15)	1	15	1 + Recursive(2, 15) 1 + 24 = 25																							
Recursive(2, 15)	2	15	2 + Recursive(4, 15) 2 + 22 = 24																							
Recursive(4, 15)	4	15	4 + Recursive(8, 15) 4 + 18 = 22																							
Recursive(8, 15)	8	15	8 + Recursive(16, 15) 8 + 10 = 18																							
Recursive(16, 15)	16	15	10																							

Question	Answer	Marks
5(b)	1 mark per bullet point to max 7 - function declaration (and end) taking two parameters and the function returns the final totalling value outside of loop and in all cases - Initialising totalling value to 0 outside of loop - Loop until Num1 >= Num2 // loop while Num1 < Num2 adding Num1 to totalling value inside the loop ... and multiplying Num1 by 2 inside a loop and storing back in Num1 After loop - Checking if Num1 > Num2 adding 10 to totalling value when true - check Num1 = Num2 adding Num1 to totalling value when true Example pseudocode: <pre> FUNCTION NonRecursive(Num1, Num2 : INTEGER) RETURNS INTEGER Value ← 0 WHILE Num1 < Num2 Value ← Value + Num1 Num1 ← Num1 * 2 ENDWHILE IF Num1 > Num2 THEN Value ← Value + 10 ELSE Value ← Value + Num1 ENDIF RETURN Value ENDFUNCTION </pre>	7

Question	Answer	Marks
6(a)	The last one in // most recent	1
6(b)(i)	1 mark for True and False in the correct place 1 for each other completed statement <pre> FUNCTION AddItemToStack(BYREF ErrorArray : ARRAY[0:99] OF Error, BYREF LastItem : INTEGER, BYVALUE Error1 : Error) RETURNS BOOLEAN IF LastItem = 99 // ErrorArray.Length - 1 THEN RETURN FALSE ELSE ErrorArray(LastItem + 1) ← Error1 LastItem ← LastItem + 1 RETURN TRUE ENDIF ENDFUNCTION </pre>	4
6(b)(ii)	1 mark per bullet point to max 3 • The function needs to change the values in ErrorArray and/or LastItem in main/where called • ... otherwise they would not be changed outside of the function // otherwise changes would only stay in the function • Error1's value does not change in the function // no changes to Error1's value need reflecting where it was called / to the original • BYVALUE stops the value being changed outside the function but BYREF changes the value where called from	3

Question	Answer	Marks
6(b)(iii)	1 mark for both return statements 1 mark for each other completed statement <pre> FUNCTION RemoveItem(ByRef ErrorArray : ARRAY[0:99] OF Error, ByRef LastItem : INTEGER) RETURNS Error DECLARE ItemToRemove : Error IF LastItem < 0 / = -1 THEN RETURN NullError ELSE ItemToRemove ← ErrorArray[LastItem] LastItem ← LastItem - 1 RETURN ItemToRemove ENDFUNCTION </pre>	3

Question	Answer	Marks
6(b)(iv)	1 mark per bullet point to max 5 • Using RemoveItem(ErrorArray, LastItem) and storing return value ... • ...checking if return value is NullError and outputting "stack empty" message if it is null • ... (if not NullError), calling Enqueue with return value ... • ... if return value is TRUE, output "added to queue" message ... • ... if return value is FALSE output "not added to queue" message <pre> PROCEDURE RunError(BYREF ErrorComplete : ARRAY[0:99] OF Error, BYREF ErrorArray : ARRAY[0:99] OF Error) DECLARE DataItem : error DataItem ← RemoveItem(ErrorArray, LastItem) IF DataItem = NullError THEN OUTPUT "Stack empty" ELSE IF Enqueue(DataItem) = True THEN OUTPUT "Item added to queue" ELSE OUTPUT "Item not added to queue" ENDIF ENDIF ENDPROCEDURE </pre>	5

Question	Answer	Marks
7(a)	1 mark per bullet point to max 5 • class header (and end where appropriate) • contents array declared of type <code>FieldObject</code> with 10 elements • size, lock and strength all private (size & lock – string, strength – integer) • constructor taking 3 parameters ... • ... setting Size, Lock and Contents at index 0/1 to parameters • ... setting strength to 100 Example program code VB.NET <pre>Public Class Box Private Size As String Private Contents(9) As FieldObject Private Lock As String Private Strength As Integer Sub New(sizep, firstContent, lockNumber) Size = sizep Lock = lockNumber Strength = 100 Contents(0) = firstContent End Sub End Class</pre> Python <pre>class Box: def __init__(self, Sizep, FirstContent, LockNumber): self.__Size = Sizep #string self.__Lock = LockNumber #string self.__Strength = 100 #integer self.__Contents[0] = FirstContent #array 10 elements of FieldObject</pre>	5

Question	Answer	Marks
7(a)	Pascal type Box = class private Size : String; Contents : array[0 .. 9] of String; Lock : String; Strength : integer; public constructor create(Sizep : String; FirstContent : String; LockNumber : string); end; constructor Box.create(Sizep : String; FirstContent : String; LockNumber : string); begin Size := Sizep; Lock := LockNumber; Strength := 100; Contents[0] := FirstContent; end; end;	

Question	Answer	Marks
7(b)	1 mark per bullet point to max 5 • Function declaration (and end) taking (string) parameter (and return Boolean) • Check if parameter matches Lock and returning true if it does • (otherwise) decrementing Strength ... • ... If Strength is < 1 / = 0, return true • ... otherwise if Strength is >= 1, return false Example program code: V.B.NET <pre>Function Unlock(Key) If Lock = Key Then Return True Else Strength = Strength - 1 If Strength <= 0 Then Return True Else Return False End If End If End Function</pre> Python <pre>def Unlock(self, Key): if self.__Lock == Key: return True else: self.__Strength = self.__Strength - 1 if self.__Strength <= 0: return True else: return False</pre>	5

Question	Answer	Marks
7(b)	Pascal function Box.Unlock(Key : String) : Boolean; begin if Lock = Key then begin Unlock := true; end else begin Strength := Strength - 1; if Strength <= 0 then begin Unlock := true; end else begin Unlock := false; end end end end; end;	

Question	Answer	Marks
7(c)	1 mark per bullet point to max 6 • procedure heading (and end where applicable) • opening progress.txt to read • read all data from file into GameData • closing file • Exception check when trying to open the file ... • ... appropriate message/other Example program code VB.NET <pre> Sub LoadGame() Dim Filename As String = "progress.txt" Dim GameData As String Try Dim ObjRead As New System.IO.StreamReader(Filename) GameData = ObjRead.ReadToEnd Console.WriteLine(GameData) ObjRead.Close() Catch Console.WriteLine("File not found") End Try End Sub </pre>	6

Question	Answer	Marks
7(c)	Python <pre>def LoadGame(): Filename = "progress.txt" try: F = open(Filename, "r") GameData = F.read() F.close() except: print("File not found")</pre> Pascal <pre>procedure LoadGame(); var Myfile : Text; GameData : String; begin try assign(Myfile, 'progress.txt'); reset(Myfile); read(Myfile, GameData); close(Myfile); except writeln('File not found'); end; end;</pre>	

Question	Answer	Marks
8	1 mark per bullet point • <code>person(X) AND has(X, black)</code> • <code>AND (has(X, moustache) OR has(X, beard))</code> Example: <code>person(X) AND has(X, black) AND (has(X, moustache) OR has(X, beard))</code>	2