



## 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.

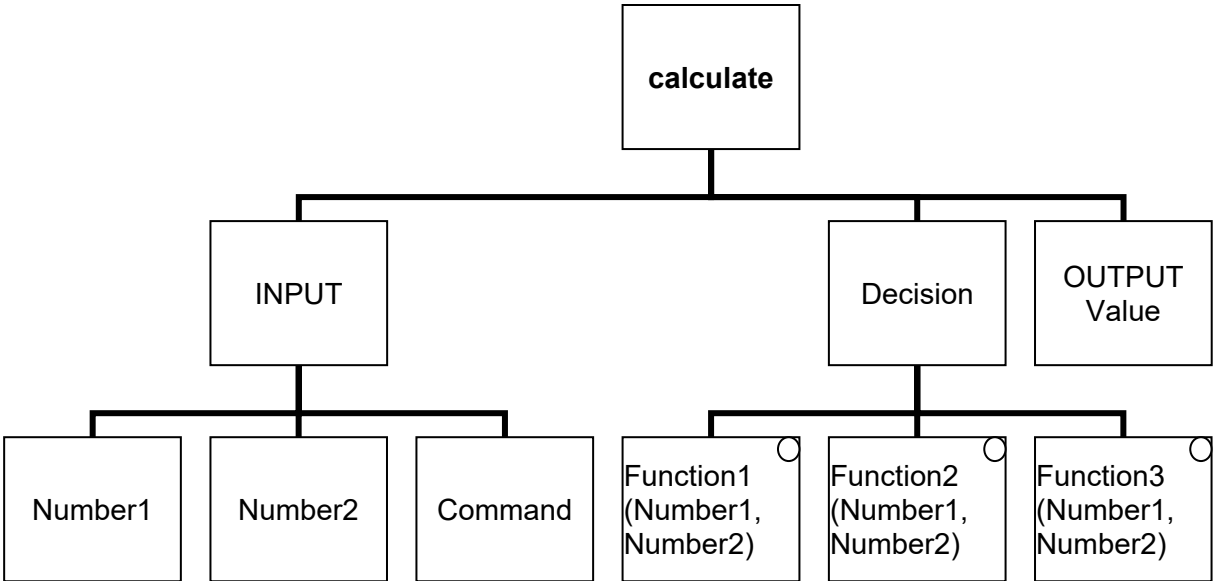
**GENERIC MARKING PRINCIPLE 5:**

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)	<p>1 mark per bullet point to max 4</p> <ul style="list-style-type: none"> <li>• Set the first element to be the sorted list</li> <li>• Store the next element in a temporary variable // store the value to be sorted in a temporary variable</li> <li>• ... compare this next element to each element in the sorted list</li> <li>• 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</li> <li>• Loop through all items from 2nd to end of array/100</li> </ul>	<b>4</b>
1(b)	<p>1 mark for each completed statement</p> <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] &gt; NumberArray[Inner + 1]         THEN           Temp ← NumberArray[Inner]           NumberArray[Inner] ← NumberArray[Inner + 1]           NumberArray[Inner + 1] ← Temp           Swap ← <b>TRUE</b>         ENDIF       Inner ← Inner + 1     UNTIL Inner = <b>Outer</b>     Outer ← Outer - 1   UNTIL Swap = <b>FALSE</b> OR Outer = 0 ENDPROCEDURE </pre>	<b>5</b>

Question	Answer	Marks
2	<p>1 mark per bullet point:</p> <ul style="list-style-type: none"> <li>• input of Number1 Number2 Command on same level under an input box</li> <li>• three functions (1, 2, 3) below e.g. decision box ...</li> <li>• ...with selection on each</li> <li>• output value at end in box below calculate</li> </ul> 	4

Question	Answer										Marks																																																																		
3	1 mark for each row <table border="1" data-bbox="338 284 1563 740" style="margin-left: auto; margin-right: auto;"> <thead> <tr> <th colspan="2"></th> <th colspan="8">Rules</th> </tr> </thead> <tbody> <tr> <td rowspan="3" style="text-align: center;"><b>Conditions</b></td> <td style="text-align: center;">One or more upper-case letters</td> <td style="text-align: center;">N</td> <td style="text-align: center;">Y</td> <td style="text-align: center;">N</td> <td style="text-align: center;">Y</td> <td style="text-align: center;">N</td> <td style="text-align: center;">Y</td> <td style="text-align: center;">N</td> <td style="text-align: center;">Y</td> </tr> <tr> <td style="text-align: center;">One or more numeric characters</td> <td style="text-align: center;">N</td> <td style="text-align: center;">N</td> <td style="text-align: center;">Y</td> <td style="text-align: center;">Y</td> <td style="text-align: center;">N</td> <td style="text-align: center;">N</td> <td style="text-align: center;">Y</td> <td style="text-align: center;">Y</td> </tr> <tr> <td style="text-align: center;">One or more symbols</td> <td style="text-align: center;">N</td> <td style="text-align: center;">N</td> <td style="text-align: center;">N</td> <td style="text-align: center;">N</td> <td style="text-align: center;">Y</td> <td style="text-align: center;">Y</td> <td style="text-align: center;">Y</td> <td style="text-align: center;">Y</td> </tr> <tr> <td rowspan="3" style="text-align: center;"><b>Actions</b></td> <td style="text-align: center;">Strong</td> <td></td> <td></td> <td></td> <td style="text-align: center;">Y</td> <td></td> <td style="text-align: center;">Y</td> <td style="text-align: center;">Y</td> <td style="text-align: center;">Y</td> </tr> <tr> <td style="text-align: center;">Medium</td> <td></td> <td style="text-align: center;">Y</td> <td style="text-align: center;">Y</td> <td></td> <td style="text-align: center;">Y</td> <td></td> <td></td> <td></td> </tr> <tr> <td style="text-align: center;">Weak</td> <td style="text-align: center;">Y</td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>												Rules								<b>Conditions</b>	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	<b>Actions</b>	Strong				Y		Y	Y	Y	Medium		Y	Y		Y				Weak	Y								<b>3</b>
		Rules																																																																											
<b>Conditions</b>	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																																																																				
<b>Actions</b>	Strong				Y		Y	Y	Y																																																																				
	Medium		Y	Y		Y																																																																							
	Weak	Y																																																																											

Question	Answer										Marks
4(a)	1 mark per bullet point <ul style="list-style-type: none"> <li>• Record declaration with identifier Node ...</li> <li>• ... all three fields declared with type integer</li> </ul> Example: <pre> TYPE Node   DECLARE LeftPointer : INTEGER   DECLARE Data : INTEGER   DECLARE RightPointer : INTEGER ENDTYPE                     </pre>										<b>2</b>

Question	Answer	Marks
4(b)	1 mark per bullet point: <ul style="list-style-type: none"> <li>• Declaration with correct identifier (Node100) of type Node</li> <li>• Assigning LeftPointer to 1 and RightPointer to 4</li> <li>• Assigning 100 to the Data</li> </ul> Example pseudocode  <pre> DECLARE Node100 : Node Node100.LeftPointer ← 1 Node100.Data ← 100 Node100.RightPointer ← 4           </pre>	<b>3</b>
4(c)(i)	To point to the start/first of the empty node/nodes	<b>1</b>
4(c)(ii)	-1 or below // 101 or above	<b>1</b>

Question	Answer	Marks
4(c)(iii)	<p>1 mark for 23 at top, with 5 below left, 100 below right                      1 mark for remaining in correct places below 5 and 100</p> <pre>                     graph TD                         23[23] --- 5[5]                         23 --- 100[100]                         5 --- E5[ ]                         5 --- 8[8]                         8 --- E8[ ]                         8 --- 9[9]                         100 --- 88[88]                         100 --- E100[ ]                         88 --- E88L[ ]                         88 --- E88R[ ]                         E100 --- E100L[ ]                         E100 --- E100R[ ]                     </pre>	2



Question	Answer	Marks
4(c)(iv)	<p>1 mark for each completed statement</p> <pre> PROCEDURE AddData(NewNode)   BinaryTree[FreePointer] ← <b>NewNode</b>   BinaryTree[FreePointer].LeftPointer ← -1   BinaryTree[FreePointer].RightPointer ← -1    DECLARE PositionFound : BOOLEAN   DECLARE PointerCounter : INTEGER   PositionFound ← <b>FALSE</b>   PointerCounter ← RootNode    WHILE NOT <b>PositionFound</b>     IF <b>NewNode.Data</b> &lt; 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 <ul style="list-style-type: none"> <li>• Return value of 25 (in space or if space left empty look at tracing)</li> <li>• Calling with 1 and 15, then 2 and 15</li> <li>• Calling with 4, then 8, then 16</li> <li>• Showing the unwinding of the return values</li> </ul> <table border="1" data-bbox="338 427 1561 951"> <thead> <tr> <th data-bbox="338 427 674 491">Function Call</th> <th data-bbox="674 427 824 491">Num1</th> <th data-bbox="824 427 956 491">Num2</th> <th data-bbox="956 427 1561 491">Return value</th> </tr> </thead> <tbody> <tr> <td data-bbox="338 491 674 587">Recursive(1, 15)</td> <td data-bbox="674 491 824 587">1</td> <td data-bbox="824 491 956 587">15</td> <td data-bbox="956 491 1561 587">1 + Recursive(2, 15) 1 + 24 = 25</td> </tr> <tr> <td data-bbox="338 587 674 683">Recursive(2, 15)</td> <td data-bbox="674 587 824 683">2</td> <td data-bbox="824 587 956 683">15</td> <td data-bbox="956 587 1561 683">2 + Recursive(4, 15) 2 + 22 = 24</td> </tr> <tr> <td data-bbox="338 683 674 778">Recursive(4, 15)</td> <td data-bbox="674 683 824 778">4</td> <td data-bbox="824 683 956 778">15</td> <td data-bbox="956 683 1561 778">4 + Recursive(8, 15) 4 + 18 = 22</td> </tr> <tr> <td data-bbox="338 778 674 874">Recursive(8, 15)</td> <td data-bbox="674 778 824 874">8</td> <td data-bbox="824 778 956 874">15</td> <td data-bbox="956 778 1561 874">8 + Recursive(16, 15) 8 + 10 = 18</td> </tr> <tr> <td data-bbox="338 874 674 951">Recursive(16, 15)</td> <td data-bbox="674 874 824 951">16</td> <td data-bbox="824 874 956 951">15</td> <td data-bbox="956 874 1561 951">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	<b>4</b>
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)	<p>1 mark per bullet point to max 7</p> <ul style="list-style-type: none"> <li>• function declaration (and end) taking two parameters <b>and</b> the function returns the final totalling value outside of loop and in all cases</li> <li>• Initialising totalling value to 0 outside of loop</li> <li>• Loop until Num1 &gt;= Num2 // loop while Num1 &lt; Num2 ...</li> <li>• ... adding Num1 to totalling value inside the loop</li> <li>• ... and multiplying Num1 by 2 inside a loop and storing back in Num1</li> </ul> <p>After loop</p> <ul style="list-style-type: none"> <li>• Checking if Num1 &gt; Num2 ...</li> <li>• ... adding 10 to totalling value when true</li> <li>• check Num1 = Num2 ...</li> <li>• ... adding Num1 to totalling value when true</li> </ul> <p>Example pseudocode:</p> <pre> FUNCTION NonRecursive(Num1, Num2 : INTEGER) RETURNS INTEGER   Value ← 0    WHILE Num1 &lt; Num2     Value ← Value + Num1     Num1 ← Num1 * 2   ENDWHILE    IF Num1 &gt; 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)	<p>1 mark for True and False in the correct place 1 for each other completed statement</p> <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)	<p>1 mark per bullet point to max 3</p> <ul style="list-style-type: none"> <li>• The function needs to change the values in <code>ErrorArray</code> and/or <code>LastItem</code> in main/where called</li> <li>• ... otherwise they would not be changed outside of the function // otherwise changes would only stay in the function</li> <li>• <code>Error1</code>'s value does not change in the function // no changes to <code>Error1</code>'s value need reflecting where it was called / to the original</li> <li>• <code>BYVALUE</code> stops the value being changed outside the function but <code>BYREF</code> changes the value where called from</li> </ul>	3

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

Question	Answer	Marks
6(b)(iv)	<p>1 mark per bullet point to max 5</p> <ul style="list-style-type: none"> <li>• Using RemoveItem(ErrorArray, LastItem) <b>and</b> storing return value ...</li> <li>• ...checking if return value is NullError <b>and</b> outputting "stack empty" message if it is null</li> <li>• ... (if not NullError), calling Enqueue with return value ...</li> <li>• ... if return value is TRUE, output "added to queue" message ...</li> <li>• ... if return value is FALSE output "not added to queue" message</li> </ul> <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   ENDPROCEDURE </pre>	<b>5</b>

Question	Answer	Marks
7(a)	<p>1 mark per bullet point to max 5</p> <ul style="list-style-type: none"> <li>• class header (and end where appropriate)</li> <li>• contents array declared of type <code>FieldObject</code> with 10 elements</li> <li>• size, lock and strength all private (size &amp; lock – string, strength – integer)</li> <li>• constructor taking 3 parameters ...</li> <li>• ... setting Size, Lock and Contents at index 0/1 to parameters</li> <li>• ... setting strength to 100</li> </ul> <p>Example program code</p> <p><b>VB.NET</b></p> <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> <p><b>Python</b></p> <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>	<b>5</b>

Question	Answer	Marks
7(a)	<pre> <b>Pascal</b> 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; </pre>	



Question	Answer	Marks
7(b)	<p>1 mark per bullet point to max 5</p> <ul style="list-style-type: none"> <li>• Function declaration (and end) taking (string) parameter (and return Boolean)</li> <li>• Check if <b>parameter</b> matches <code>Lock</code> <b>and</b> returning true if it does</li> <li>• (otherwise) decrementing <code>Strength</code> ...</li> <li>• ... If <code>Strength</code> is <code>&lt; 1 / = 0</code>, return true</li> <li>• ... otherwise if <code>Strength</code> is <code>&gt;= 1</code>, return false</li> </ul> <p>Example program code:</p> <p><b>V.B.NET</b></p> <pre>Function Unlock(Key)   If Lock = Key Then     Return True   Else     Strength = Strength - 1     If Strength &lt;= 0 Then       Return True     Else       Return False     End If   End If End Function</pre> <p><b>Python</b></p> <pre>def Unlock(self, Key):   if self.__Lock == Key:     return True   else:     self.__Strength = self.__Strength - 1     if self.__Strength &lt;= 0:       return True     else:       return False</pre>	5

<b>Question</b>	<b>Answer</b>	<b>Marks</b>
7(b)	<b>Pascal</b> 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;	

Question	Answer	Marks
7(c)	<p>1 mark per bullet point to max 6</p> <ul style="list-style-type: none"> <li>• procedure heading (and end where applicable)</li> <li>• opening progress.txt to read</li> <li>• read all data from file into GameData</li> <li>• closing file</li> <li>• Exception check when trying to open the file ...</li> <li>• ... appropriate message/other</li> </ul> <p>Example program code</p> <p><b>VB.NET</b></p> <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>	<b>6</b>

Question	Answer	Marks
7(c)	<p><b>Python</b></p> <pre>def LoadGame():     Filename = "progress.txt"     try:         F = open(Filename, "r")         GameData = F.read()         F.close()      except:         print("File not found")</pre> <p><b>Pascal</b></p> <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>	

<b>Question</b>	<b>Answer</b>	<b>Marks</b>
8	1 mark per bullet point <ul style="list-style-type: none"><li>• <code>person(X) AND has(X, black)</code></li><li>• <code>AND (has(X, moustache) OR has(X, beard))</code></li></ul> Example: <code>person(X) AND has(X, black) AND (has(X, moustache) OR has(X, beard))</code>	<b>2</b>