



## Cambridge International AS & A Level

---

COMPUTER SCIENCE

9608/43

Paper 4 Written Paper

October/November 2020

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 2020 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 **15** 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.

https://xtremepape.rs/

Question	Answer																				Marks	
1	A																				<b>5</b>	
	B																					
	C																					
	D																					
	E																					
	F																					
	G																					
	H																					
	I																					
	J																					
	K																					
	Week number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19		20
<ul style="list-style-type: none"> <li>• A(1) and B(3) following A</li> <li>• C(6) following B and D(4) following B</li> <li>• E(2) following D, F(2) following E, G(1) following F</li> <li>• H(2) following G, I(2) following G, J(3) following G</li> <li>• K(4) following H</li> </ul>																						

**PUBLISHED**

Question	Answer					Marks																														
2(a)	<table border="1"> <thead> <tr> <th data-bbox="365 213 1308 280">Statement</th> <th data-bbox="1308 213 1496 280">Integration</th> <th data-bbox="1496 213 1688 280">Acceptance</th> <th data-bbox="1688 213 1805 280">Alpha</th> <th data-bbox="1805 213 1912 280">Beta</th> </tr> </thead> <tbody> <tr> <td data-bbox="365 280 1308 344">Software is tested in-house by dedicated testers</td> <td data-bbox="1308 280 1496 344">✓</td> <td data-bbox="1496 280 1688 344"></td> <td data-bbox="1688 280 1805 344">✓</td> <td data-bbox="1805 280 1912 344">(✓)</td> </tr> <tr> <td data-bbox="365 344 1308 408">Software is tested by the client before it is signed-off</td> <td data-bbox="1308 344 1496 408"></td> <td data-bbox="1496 344 1688 408">✓</td> <td data-bbox="1688 344 1805 408"></td> <td data-bbox="1805 344 1912 408"></td> </tr> <tr> <td data-bbox="365 408 1308 504">Software is tested by combining modules that have previously been tested to check they work as expected.</td> <td data-bbox="1308 408 1496 504">✓</td> <td data-bbox="1496 408 1688 504"></td> <td data-bbox="1688 408 1805 504"></td> <td data-bbox="1805 408 1912 504"></td> </tr> <tr> <td data-bbox="365 504 1308 576">Software is tested using normal, abnormal and boundary data.</td> <td data-bbox="1308 504 1496 576">✓</td> <td data-bbox="1496 504 1688 576">✓</td> <td data-bbox="1688 504 1805 576">✓</td> <td data-bbox="1805 504 1912 576">(✓)</td> </tr> <tr> <td data-bbox="365 576 1308 676">Software is tested by releasing it to selected customers, who test it in normal circumstances</td> <td data-bbox="1308 576 1496 676"></td> <td data-bbox="1496 576 1688 676"></td> <td data-bbox="1688 576 1805 676"></td> <td data-bbox="1805 576 1912 676">✓</td> </tr> </tbody> </table>	Statement	Integration	Acceptance	Alpha	Beta	Software is tested in-house by dedicated testers	✓		✓	(✓)	Software is tested by the client before it is signed-off		✓			Software is tested by combining modules that have previously been tested to check they work as expected.	✓				Software is tested using normal, abnormal and boundary data.	✓	✓	✓	(✓)	Software is tested by releasing it to selected customers, who test it in normal circumstances				✓					4
Statement	Integration	Acceptance	Alpha	Beta																																
Software is tested in-house by dedicated testers	✓		✓	(✓)																																
Software is tested by the client before it is signed-off		✓																																		
Software is tested by combining modules that have previously been tested to check they work as expected.	✓																																			
Software is tested using normal, abnormal and boundary data.	✓	✓	✓	(✓)																																
Software is tested by releasing it to selected customers, who test it in normal circumstances				✓																																
2(b)	One from: <ul style="list-style-type: none"> <li>• dry run</li> <li>• walkthrough</li> <li>• white-box</li> <li>• black-box</li> </ul>					1																														

**PUBLISHED**

Question	Answer	Marks
3(a)	<ul style="list-style-type: none"> <li>• Correct header and close (where applicable) with no parameters ...</li> <li>• ...Correct values assigned to correct identifiers</li> </ul> <p><b>PYTHON</b></p> <pre>def __init__(self):     self.__PlayerID = "PL12a3"     self.__Name = ""     self.__Score = 0</pre> <p><b>PASCAL</b></p> <pre>Constructor PuzzlePlayer.Create(); begin     PlayerID := 'PL12a3';     Name := '';     Score := 0; end;</pre> <p><b>VB</b></p> <pre>Public Sub New()     PlayerID = "PL12a3"     Name = ""     Score = 0 End Sub</pre>	<b>2</b>

**PUBLISHED**

Question	Answer	Marks
3(b)	<ul style="list-style-type: none"> <li>• Correct <b>function</b> header and close (where applicable) no parameter (if returns value must be string)</li> <li>• Returns correct value <code>PlayerID</code></li> </ul> <p><b>PYTHON</b></p> <pre>def GetPlayerID(self):     return(self.__PlayerID)</pre> <p><b>PASCAL</b></p> <pre>Function GetPlayerID() : String; Begin     GetPlayerID := PlayerID; End;</pre> <p><b>VB</b></p> <pre>public function GetPlayerID()     return(PlayerID) End Function</pre>	<b>2</b>

Question	Answer	Marks
3(c)	<ul style="list-style-type: none"> <li>• <b>Function</b> header (and close), value passed as parameter (returning Boolean/String if present)</li> <li>• Checks parameter is 6 characters in length</li> <li>• <b>AND</b> checks the first two characters in parameter are "PL"</li> <li>• ...Returns true if parameter is valid and stores in PlayerID</li> <li>• ...Returns false if parameter is not valid (and stores in PlayerID or not)</li> </ul> <pre> FUNCTION SetPlayerID(NewPlayerID) RETURNS BOOLEAN   DECLARE Valid : BOOLEAN   Valid ← TRUE   IF Length(NewPlayerID) = 6 AND Substring(NewPlayerID,0,2) = "PL"     THEN       PlayerID ← NewPlayerID     ELSE       Valid ← FALSE   ENDIF   RETURN Valid ENDFUNCTION </pre>	<b>5</b>
3(d)(i)	<ul style="list-style-type: none"> <li>• A class contains objects/instances of another class</li> <li>• Quiz class has objects of type Question class // The objects/items in the array QuizQs have the attributes/methods of the class Question // The array QuizQs is of data type class Question</li> </ul>	<b>2</b>
3(d)(ii)	<p>e.g.</p> <ul style="list-style-type: none"> <li>• Inheritance</li> <li>• A child class can use attributes/methods from the parent class</li> <li>• Polymorphism</li> <li>• A child class can overwrite the methods of the parent class</li> </ul>	<b>2</b>
3(d)(iii)	<ul style="list-style-type: none"> <li>• Correct identifier, 100 elements, type Quiz and clearly array (i.e. brackets)</li> </ul> <pre> DECLARE QuizBank : ARRAY[0:99] OF Quiz </pre>	<b>1</b>



**PUBLISHED**

Question	Answer	Marks
3(d)(iv)	<ul style="list-style-type: none"> <li>Instance of object and assignment to first element of array (0 or 1)</li> <li>Correct parameters (in any order)</li> </ul> <p><b>PYTHON</b> QuizBank[0] = Quiz("Famous people", "Low", 10)</p> <p><b>PASCAL</b> QuizBank[0] := Quiz.Create('Famous people', 'Low', 10);</p> <p><b>VB</b> QuizBank[0] = New Quiz ("Famous people", "Low", 10)</p>	<b>2</b>

Question	Answer	Marks
4(a)	<ul style="list-style-type: none"> <li>type(camembert)</li> <li>soft(camembert)</li> <li>strong_smell(camembert)</li> <li>origin(camembert, france)</li> </ul>	<b>4</b>
4(b)	<ul style="list-style-type: none"> <li>type(X)</li> <li>AND country(Y)</li> <li>AND NOT</li> <li>soft(X)</li> </ul> <p>IF type (X) AND country (Y) AND NOT soft(X)</p>	<b>4</b>

**PUBLISHED**

Question	Answer	Marks
5(a)	<p>Four from:</p> <ul style="list-style-type: none"> <li>• Uses a sorted <b>and</b> unsorted list</li> <li>• Takes first value and makes it sorted list // compare second item to first item</li> <li>• Find location of next value in the sorted list</li> <li>• ...description of suitable method (e.g. switching values, taking value out, comparing with sorted values)</li> <li>• Insert item in correct position in sorted list</li> <li>• Repeat until all items are in the sorted list (dependent on suitable method)</li> </ul>	<b>4</b>
5(b)	<pre> Counter ← NumberOfItems - 2 REPEAT   DataSwapped ← FALSE   FOR CurrentValue ← 0 TO Counter // NumberOfItems - 2     IF DataList[CurrentValue] &gt; DataList[CurrentValue + 1]       THEN         ValueTemp ← DataList[CurrentValue]         DataList[CurrentValue] ← DataList[CurrentValue + 1]         DataList[CurrentValue + 1] ← ValueTemp         DataSwapped ← TRUE       ENDFOR     UNTIL DataSwapped = FALSE </pre>	<b>5</b>

Question	Answer	Marks
6(a)	<ul style="list-style-type: none"> <li>• A–B–E</li> <li>• E–C–D with D null pointer</li> </ul>	<b>2</b>
6(b)	It indicates the end of the list // it doesn't point anywhere/to any data/to another node	<b>1</b>

Question	Answer	Marks
6(c)(i)	<pre>FUNCTION FindValue(Value : INTEGER) RETURNS INTEGER   DECLARE CurrentPointer : INTEGER   CurrentPointer ← StartPointer   WHILE <b>CurrentPointer</b> &lt;&gt; NULL AND LinkedList[CurrentPointer].Data &lt;&gt; <b>Value</b>     CurrentPointer ← LinkedList[<b>CurrentPointer</b>].Pointer   ENDWHILE   IF LinkedList[CurrentPointer].Data = Value     THEN       RETURN <b>CurrentPointer</b>     ELSE       RETURN -1   ENDIF ENDFUNCTION</pre>	<b>6</b>

**PUBLISHED**

Question	Answer	Marks
6(c)(ii)	<p>One mark per bullet point to max 7</p> <ul style="list-style-type: none"> <li>• <b>Function</b> header, taking parameter (and returning Boolean)</li> <li>• Assign a new pointer to StartPointer</li> <li>• Iterate/recursive calls through nodes correctly updating current pointer</li> <li>• Checking for empty list <b>and</b> returning FALSE</li> <li>• Checking if end of list ...</li> <li>• ... check data in last node</li> <li>• Checking if data found...</li> <li>• ... set pointer of found node to NULL (return to free chain)</li> <li>• ... if found update previous node pointer to NULL</li> <li>• ... return TRUE</li> <li>• If end of list <b>and</b> not found then return FALSE</li> </ul> <pre> FUNCTION DeleteNode(NodeData : STRING) RETURNS BOOLEAN   IF StartPointer = NULL     THEN       RETURN FALSE     ELSE       CurrentPointer ← StartPointer       IF LinkedList[CurrentPointer].Data = NodeData         THEN           StartPointer ← LinkedList[CurrentPointer].Pointer           RETURN TRUE         ELSE           PreviousPointer ← CurrentPointer           WHILE CurrentPointer &lt;&gt; NULL AND             LinkedList[CurrentPointer].Data &lt;&gt; NodeData             PreviousPointer ← CurrentPointer             CurrentPointer ← LinkedList[CurrentPointer].Pointer           ENDWHILE </pre>	<b>7</b>

Question	Answer	Marks
6(c)(ii)	<pre> IF CurrentPointer = NULL   THEN     IF LinkedList [CurrentPointer].Data = NodeData       THEN         LinkedList [PreviousPointer].Pointer ← NULL         RETURN TRUE       ELSE         RETURN FALSE       ENDIF     ELSE       IF LinkedList [CurrentPointer].Data = NodeData         THEN           LinkedList [PreviousPointer].Pointer ←             LinkedList [CurrentPointer].Pointer           LinkedList [CurrentPointer].Pointer ← NULL           RETURN TRUE         ENDIF       ENDIF     ENDIF   ENDIF ENDIF ENDFUNCTION </pre>	

Question	Answer					Marks
7	Label	Op Code	Operand	Comment		8
		LDR	#0			
		LDM	#0			
		STO	LENGTH	// initialise LENGTH to 0		
	LOOP:	IN				
		CMP	EXCLAMATIO N	// is character = EXCLAMATION ('!')?	[1]	
		JPE	OUTPUT	// if true, jump to OUTPUT	[1]	
		STX	USERNAME	// store character in USERNAME + contents of IX	[1]	
		INC	IX	// increment Index Register		
		LDD	LENGTH			
		INC	ACC	// increment LENGTH	[1]	
		STO	LENGTH			
		CMP	MAX	// is LENGTH = MAX ?	[1]	
		JPN	LOOP	// if FALSE, jump to LOOP	[1]	
	OUTPUT:	LDR	#0			
		LDM	#0			
		STO	COUNT	// initialise COUNT to 0	[1]	
		LDX	USERNAME			
		OUT				
		INC	IX			
		LDD	COUNT			
		INC	ACC	// increment COUNT		
		STO	COUNT			
		CMP	LENGTH	// is COUNT = LENGTH ?	[1]	
	JPN	OUTPUT				
	END		// end program			
LENGTH:						
EXCLAMATION:	B0010001					
MAX:	8					
COUNT:						
USERNAME:						

Question	Answer	Marks																																										
8(a)	<ul style="list-style-type: none"> <li>• 8 ...</li> <li>• ...it calls itself</li> </ul>	<b>2</b>																																										
8(b)	<p>1 mark each:</p> <ul style="list-style-type: none"> <li>• Final return value = 5</li> <li>• Output column</li> <li>• Return value column</li> <li>• Value 1 and Value 2 columns</li> <li>• Temp column</li> </ul> <table border="1" data-bbox="365 598 1485 1058" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th>Value1</th> <th>Value2</th> <th>Temp</th> <th>EndValue</th> <th>OUTPUT</th> <th>Return Value</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>1</td> <td>1</td> <td>12</td> <td>1</td> <td>5</td> </tr> <tr> <td>2</td> <td></td> <td></td> <td></td> <td>2</td> <td>4</td> </tr> <tr> <td>3</td> <td>2</td> <td>2</td> <td></td> <td>3</td> <td>3</td> </tr> <tr> <td>5</td> <td>3</td> <td>3</td> <td></td> <td>5</td> <td>2</td> </tr> <tr> <td>8</td> <td>5</td> <td>5</td> <td></td> <td>8</td> <td>1</td> </tr> <tr> <td>13</td> <td>8</td> <td></td> <td></td> <td>13</td> <td>0</td> </tr> </tbody> </table>	Value1	Value2	Temp	EndValue	OUTPUT	Return Value	1	1	1	12	1	5	2				2	4	3	2	2		3	3	5	3	3		5	2	8	5	5		8	1	13	8			13	0	<b>5</b>
Value1	Value2	Temp	EndValue	OUTPUT	Return Value																																							
1	1	1	12	1	5																																							
2				2	4																																							
3	2	2		3	3																																							
5	3	3		5	2																																							
8	5	5		8	1																																							
13	8			13	0																																							
8(c)	To output/find a value that is the addition of the two previous values // (output) Fibonacci sequence	<b>1</b>																																										