



## Cambridge International AS & A Level

---

COMPUTER SCIENCE

9618/22

Paper 2 Problem Solving & Programming

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 **10** printed pages.

**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)	The process involves: 1 Breaking down a problem / task into sub problems / steps / smaller parts 2 In order to explain / understand // easier to solve the problem 3 Leading to the concept of program modules // assigning problem parts to teams  <b>Max 2</b>	<b>2</b>										
1(b)	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th colspan="2" style="text-align: center;">Answer</th> </tr> </thead> <tbody> <tr> <td style="width: 70%;">The number of dimensions of <code>ThisArray</code></td> <td style="text-align: center;">1</td> </tr> <tr> <td>The technical terms for minimum and maximum values that variable <code>n</code> may take</td> <td style="text-align: center;"><b>Lower bound, upper bound</b></td> </tr> <tr> <td>The technical term for the variable <code>n</code> in the pseudocode expression.</td> <td style="text-align: center;"><b>Index / Subscript</b></td> </tr> </tbody> </table> <p>One mark per row</p>	Answer		The number of dimensions of <code>ThisArray</code>	1	The technical terms for minimum and maximum values that variable <code>n</code> may take	<b>Lower bound, upper bound</b>	The technical term for the variable <code>n</code> in the pseudocode expression.	<b>Index / Subscript</b>	<b>3</b>		
Answer												
The number of dimensions of <code>ThisArray</code>	1											
The technical terms for minimum and maximum values that variable <code>n</code> may take	<b>Lower bound, upper bound</b>											
The technical term for the variable <code>n</code> in the pseudocode expression.	<b>Index / Subscript</b>											
1(c)	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 60%;">Expression</th> <th>Evaluates to</th> </tr> </thead> <tbody> <tr> <td><code>ASC('C')</code></td> <td style="text-align: center;">67</td> </tr> <tr> <td><code>2 * STR_TO_NUM("27")</code></td> <td style="text-align: center;">54</td> </tr> <tr> <td><code>INT(27 / 2)</code></td> <td style="text-align: center;">13</td> </tr> <tr> <td><code>"Sub" &amp; MID("Abstraction", 4, 5)</code></td> <td style="text-align: center;">"Subtract"</td> </tr> </tbody> </table> <p>One mark per row</p> <p>Function names must be exactly as shown</p>	Expression	Evaluates to	<code>ASC('C')</code>	67	<code>2 * STR_TO_NUM("27")</code>	54	<code>INT(27 / 2)</code>	13	<code>"Sub" &amp; MID("Abstraction", 4, 5)</code>	"Subtract"	<b>4</b>
Expression	Evaluates to											
<code>ASC('C')</code>	67											
<code>2 * STR_TO_NUM("27")</code>	54											
<code>INT(27 / 2)</code>	13											
<code>"Sub" &amp; MID("Abstraction", 4, 5)</code>	"Subtract"											
1(d)	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 65%;">Expression</th> <th>Evaluates to</th> </tr> </thead> <tbody> <tr> <td><code>PressureOK AND HiFlow</code></td> <td style="text-align: center;">FALSE</td> </tr> <tr> <td><code>PumpOn OR PressureOK</code></td> <td style="text-align: center;">TRUE</td> </tr> <tr> <td><code>NOT PumpOn OR (PressureOK AND NOT HiFlow)</code></td> <td style="text-align: center;">TRUE</td> </tr> <tr> <td><code>NOT (PumpOn OR PressureOK) AND NOT HiFlow</code></td> <td style="text-align: center;">FALSE</td> </tr> </tbody> </table> <p>1 mark for any two rows correct 2 marks for all rows correct.</p>	Expression	Evaluates to	<code>PressureOK AND HiFlow</code>	FALSE	<code>PumpOn OR PressureOK</code>	TRUE	<code>NOT PumpOn OR (PressureOK AND NOT HiFlow)</code>	TRUE	<code>NOT (PumpOn OR PressureOK) AND NOT HiFlow</code>	FALSE	<b>2</b>
Expression	Evaluates to											
<code>PressureOK AND HiFlow</code>	FALSE											
<code>PumpOn OR PressureOK</code>	TRUE											
<code>NOT PumpOn OR (PressureOK AND NOT HiFlow)</code>	TRUE											
<code>NOT (PumpOn OR PressureOK) AND NOT HiFlow</code>	FALSE											

Question	Answer	Marks
2(a)	<pre> graph TD     Start([START]) --&gt; Set1[Set PosCount, NegCount to 0]     Set1 --&gt; Set2[Set PosTotal, NegTotal to 0]     Set2 --&gt; Input[/INPUT NextNum/]     Input --&gt; IsZero{Is NextNum = 0?}     IsZero -- YES --&gt; Output1[/OUTPUT PosCount, NegCount/]     Output1 --&gt; Output2[/OUTPUT PosTotal, NegTotal/]     Output2 --&gt; End([END])     IsZero -- NO --&gt; IsPos{Is NextNum &gt; 0?}     IsPos -- YES --&gt; IncPos[Increment PosCount]     IncPos --&gt; SetPos[Set PosTotal to PosTotal + NextNum]     IsPos -- NO --&gt; IncNeg[Increment NegCount]     IncNeg --&gt; SetNeg[Set NegTotal to NegTotal + NextNum]     SetPos --&gt; LoopBack     SetNeg --&gt; LoopBack     Set1 -.-&gt; LoopBack     Set2 -.-&gt; LoopBack     Input -.-&gt; LoopBack     </pre> <p>One mark for each outlined group.</p> <p>Note:</p> <ul style="list-style-type: none"> <li>Sum and increment steps (bottom-right rectangles) may be in reverse order in which case sum group will have two output lines</li> <li>Max 4 for non-working solution</li> </ul>	5

Question	Answer	Marks
2(b)(i)	One mark for each:  Life cycle method: Iterative // Rapid Application Development (RAD)  Reason: Provides a working model / prototype at an early stage for the principal to approve / review	<b>2</b>
2(b)(ii)	Decisions will be made regarding: <ul style="list-style-type: none"> <li>• Data structures</li> <li>• Algorithms / flowcharts / pseudocode</li> <li>• Program structure (modules) / use of library routines / module - team allocation</li> <li>• User interface // Web-page layout / content (for given scenario)</li> <li>• Testing method / plan</li> <li>• Choice of programming language / program environment</li> </ul> <b>Max 3</b>	<b>3</b>

Question	Answer	Marks
3(a)(i)	Pseudocode:  <pre> TYPE Student   DECLARE StudentID : STRING   DECLARE Email : STRING   DECLARE Club_1 : INTEGER   DECLARE Club_2 : INTEGER   DECLARE Club_3 : INTEGER ENDTYPE </pre> Mark as follows: <ul style="list-style-type: none"> <li>• One mark for TYPE and ENDTYPE</li> <li>• One mark for StudentID <b>and</b> Email fields as STRING</li> <li>• One mark for all Club fields as INTEGER</li> </ul>	<b>3</b>
3(a)(ii)	<u>DECLARE Membership : ARRAY [1:3000] OF Student</u>  One mark per underlined phrase	<b>2</b>
3(a)(iii)	One mark for any one of: <ul style="list-style-type: none"> <li>• Assign a value (of the correct data type) outside the normal range to one of the fields</li> <li>• Assign an empty string to the StudentID field / Email field</li> <li>• or value out of range to any club field</li> </ul>	<b>1</b>
3(a)(iv)	A number outside the range 1 to 99	<b>1</b>

Question	Answer	Marks
3(b)	<pre>PROCEDURE GetIDs()   DECLARE Index : INTEGER   DECLARE ThisClub, Count : INTEGER    OUTPUT "Please Input Club Number: "   INPUT ThisClub    Count ← 0    FOR Index ← 1 TO 3000     IF Membership[Index].Club_1 = ThisClub OR ___       Membership[Index].Club_2 = ThisClub OR ___       Membership[Index].Club_3 = ThisClub THEN       Count ← Count + 1       OUTPUT Membership[Index].StudentID     ENDIF   NEXT Index    OUTPUT "There are ", Count, " Students in the club" ENDPROCEDURE</pre> <p>Mark as follows:</p> <ol style="list-style-type: none"> <li>1 Declare and initialise Count</li> <li>2 Prompt and Input club number <b>before the loop</b></li> <li>3 Loop through 3000 elements</li> <li>4 Compare <b>one</b> club field with number input</li> <li>5 Compare <b>all</b> Club fields with number input</li> <li>6 If number found, OUTPUT of StudentID field <b>and</b> increment Count</li> <li>7 Final OUTPUT of Count <b>outside the loop</b></li> </ol> <p>Note:</p> <p>Max 6 if procedure heading and ending missing or incorrect (but allow array as parameter)</p>	7

Question	Answer	Marks												
4(a)	<table border="1"> <thead> <tr> <th colspan="2">Answer</th> </tr> </thead> <tbody> <tr> <td>A line number containing a variable being incremented</td> <td>19 / 21 / 23</td> </tr> <tr> <td>The type of loop structure</td> <td>pre-condition</td> </tr> <tr> <td>The number of functions used</td> <td>3</td> </tr> <tr> <td>The number of parameters passed to function STR_TO_NUM()</td> <td>1</td> </tr> <tr> <td>The name of a procedure other than Check()</td> <td>Result</td> </tr> </tbody> </table>	Answer		A line number containing a variable being incremented	19 / 21 / 23	The type of loop structure	pre-condition	The number of functions used	3	The number of parameters passed to function STR_TO_NUM()	1	The name of a procedure other than Check()	Result	5
Answer														
A line number containing a variable being incremented	19 / 21 / 23													
The type of loop structure	pre-condition													
The number of functions used	3													
The number of parameters passed to function STR_TO_NUM()	1													
The name of a procedure other than Check()	Result													

Question	Answer	Marks
4(b)	One mark per point: <ul style="list-style-type: none"><li>• Meaningful variable names</li><li>• Indentation / white space / blank lines</li><li>• Capitalisation of keywords</li></ul>	<b>3</b>
4(c)(i)	One mark per point:  Structure: A count-controlled loop  Justification: The number of iterations is known // repeats for the length of <code>InString</code>	<b>2</b>
4(c)(ii)	15, 23  One mark for <b>both</b> line numbers	<b>1</b>

Question	Answer	Marks
5(a)	<pre> PROCEDURE MakeNewFile(OldFile, NewFile, Status : STRING)   DECLARE Line1, Line2, Line3 : STRING   DECLARE NumCopied, NumRecs : INTEGER    NumRecs ← 0   NumCopied ← 0    OPENFILE OldFile FOR READ   OPENFILE NewFile FOR WRITE    WHILE NOT EOF(OldFile)     READFILE OldFile, Line1     READFILE OldFile, Line2     READFILE OldFile, Line3     NumRecs ← NumRecs + 1     IF Line3 &lt;&gt; Status THEN       WRITEFILE NewFile, Line1       WRITEFILE NewFile, Line2       WRITEFILE NewFile, Line3       NumCopied ← NumCopied + 1     ENDIF   ENDWHILE    OUTPUT "File " , OldFile , " contained " , NumRecs , __     " employee details"   OUTPUT NumCopied , " employee sets of details were __     written to file", NewFile    CLOSEFILE OldFile   CLOSEFILE NewFile ENDPROCEDURE </pre> <p>Mark as follows:</p> <ol style="list-style-type: none"> <li>1 Procedure heading and ending, including parameters</li> <li>2 OPEN OldFile for READ and NewFile for WRITE and subsequently CLOSE both files</li> <li>3 Conditional loop until EOF(OldFile)</li> <li>4 Read three lines from OldFile <b>in a loop</b></li> <li>5 Compare 3<sup>rd</sup> line read with Status parameter <b>and</b> if not equal write 3 lines to NewFile <b>in a loop</b></li> <li>6 Count number of sets read and those written <b>in a loop</b></li> <li>7 Final output including both counts and file names with suitable text <b>after a loop</b></li> </ol> <p>Note: MP6: Both counts must have been declared and initialised</p>	<b>7</b>
5(b)(i)	Store all three items on one line	<b>1</b>



Question	Answer	Marks
5(b)(ii)	One mark per point:  Advantage: Fewer file operations required  Disadvantage: Algorithm to combine / extract individual data items is more complex	2

Question	Answer	Marks
6(a)	<pre> FUNCTION FirstRowSet () RETURNS INTEGER   DECLARE Row, Col : INTEGER   DECLARE Found : BOOLEAN    // array is 1280 x 800   Row ← 1    Found ← FALSE   WHILE Row &lt;= 800 AND Found = FALSE // top to bottom     Col ← 1     WHILE Col &lt;= 1280 AND Found = FALSE // left to right       IF Screen[Row,Col] = 1 THEN         Found ← TRUE // end function as soon as first                       // found       ENDIF       Col ← Col + 1     ENDWHILE     Row ← Row + 1   ENDWHILE    IF Found = FALSE THEN // nothing found     Row ← 0   ENDIF   RETURN Row - 1 ENDFUNCTION </pre> <p>Mark as follows:</p> <ol style="list-style-type: none"> <li>1 Function heading and ending <b>and</b> return type</li> <li>2 (Conditional) outer loop 1 to 800 (row)</li> <li>3 (Conditional) inner loop 1 to 1280 // 1280 to 1 (column)</li> <li>4 Reference <code>Screen</code> element and test for = 1 // &lt;&gt; 0</li> <li>5 and if true save row number <b>and</b> exit loops</li> <li>6 Increment index variables in both inner <b>and</b> outer loop</li> <li>7 Return Row number or -1, following a reasonable attempt</li> </ol>	7
6(b)	<p>One mark for:</p> <ul style="list-style-type: none"> <li>• (A flag is used to) exit the loops // iteration is terminated</li> <li>• as soon as a <code>Screen</code> element with value 1 is found</li> </ul>	2

Question	Answer	Marks
6(c)(i)	<p>One mark for:</p> <ul style="list-style-type: none"> <li>Parameter(s) need to be passed to the module to identify the type of search</li> <li>Search algorithm is controlled by (global) variables / parameters</li> </ul> <p>Alternative:</p> <ul style="list-style-type: none"> <li>The search algorithms from the original modules are included in the new module</li> <li>The new module needs to return / store the four values (the results of the four searches)</li> </ul>	2
6(c)(ii)	<p>One mark for advantage and one for disadvantage:</p> <p>Advantage: (max 1)</p> <ul style="list-style-type: none"> <li>Only have to change one module if specification changes</li> <li>Less repetitive code / fewer lines of code</li> <li>Aids re-usability</li> </ul> <p>Disadvantage: (max 1)</p> <ul style="list-style-type: none"> <li>Single module more complex / more error prone / more difficult to debug ...</li> <li>Single module cannot be split among programmers / teams</li> </ul> <p><b>Max 2</b></p>	2
6(d)	<pre>PROCEDURE GetCentre ()   DECLARE StartRow, EndRow, StartCol, EndCol : INTEGER    StartRow ← FirstRowSet ()   IF StartRow = -1 THEN     CentreRow ← -1 // no 'touch' detected   ELSE     EndRow ← LastRowSet ()     StartCol ← FirstColSet ()     EndCol ← LastColSet ()     CentreRow ← INT((StartRow + EndRow)/2)     CentreCol ← INT((StartCol + EndCol)/2)   ENDIF ENDPROCEDURE</pre> <p>Mark as follows:</p> <ol style="list-style-type: none"> <li>Call &lt;any Set function&gt; and check for -1 // check for no element set</li> <li>...and if so set CentreRow to -1</li> <li>Call all 4 Set functions to get 'extremity' values</li> <li>Calculate centre row and centre column</li> <li>Use of INT() function or DIV operator on values from MP4</li> <li>Assign calculated values to CentreRow and CentreCol</li> </ol> <p>Note: Max 5 if procedure heading and ending missing or incorrect (ignore array if passed as a parameter) <b>or</b> any local variables are undefined or of incorrect type</p>	6