CAMBRIDGE
International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/21**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2024**

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 May/June 2024 series for most Cambridge IGCSE, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

 [Turn over

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 descriptions 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:

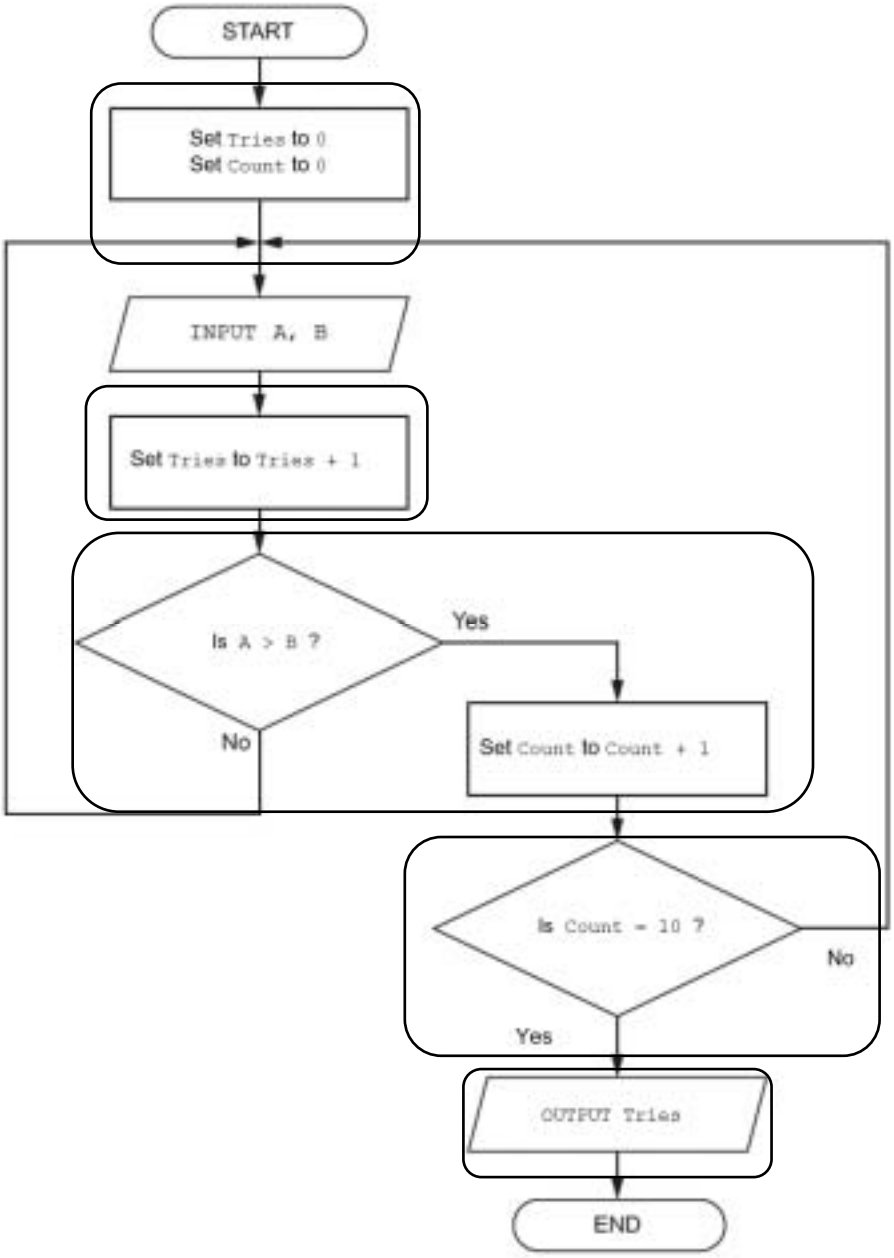
Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

Mark scheme abbreviations

/	separates alternative words / phrases within a marking point
//	separates alternative answers within a marking point
underline	actual word given must be used by candidate (grammatical variants accepted)
max	indicates the maximum number of marks that can be awarded
( )	the word / phrase in brackets is not required, but sets the context

Note: No marks are awarded for using brand names of software packages or hardware.

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>1(a)</td><td>
<table>
<tr><th>Assignment statement</th><th>Data type</th></tr>
<tr><td>A ← LEFT(MyName, 1)</td><td>CHAR / STRING</td></tr>
<tr><td>B ← Total * 2</td><td>INTEGER / REAL</td></tr>
<tr><td>C ← INT(ItemCost) / 3</td><td>REAL</td></tr>
<tr><td>D ← "Odd OR Even"</td><td>STRING</td></tr>
</table>
One mark per row</td><td>4</td></tr>
<tr><td>1(b)</td><td>
<table>
<tr><th>Expression</th><th>Evaluates to</th></tr>
<tr><td>Tries < 10 AND NOT Sorted</td><td>TRUE</td></tr>
<tr><td>Tries MOD 4</td><td>1</td></tr>
<tr><td>TO_LOWER(MID(ID, 3, 1))</td><td>'a' // "a"</td></tr>
<tr><td>LENGTH(ID & "xx") >= Tries</td><td>TRUE</td></tr>
</table>
One mark per row</td><td>4</td></tr>
<tr><td>1(c)(i)</td><td>The names do not reflect / indicate the purpose of the variable // the names are not meaningful</td><td>1</td></tr>
<tr><td>1(c)(ii)</td><td>They make the program more difficult to understand / debug / maintain</td><td>1</td></tr>
<tr><td>1(c)(iii)</td><td>One mark from:
• Indentation / use of white space
• Capitalisation of keywords
• Use of comments
• Use of modular programming
• Use of local variables
Note: max 1 mark</td><td>1</td></tr>
</table>

Question	Answer	Marks
2(a)	 One mark per outlined region: 1 Initialise both counts 2 Increment `Tries` every time a pair is input 3 Compare A > B **and** increment `Count` if TRUE 4 Test for Count = 10 (10th time A > B) – MUST include Yes / No labels 5 If so output `Tries`, otherwise loop	**5**
2(b)	One mark per point: 1 A (variable of type) string will be input // by example e.g. “67,72” 2 A special / identified character would need to be used to separate each numeric value // all numbers are fixed length	**2**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>3(a)</td><td>One mark per point:

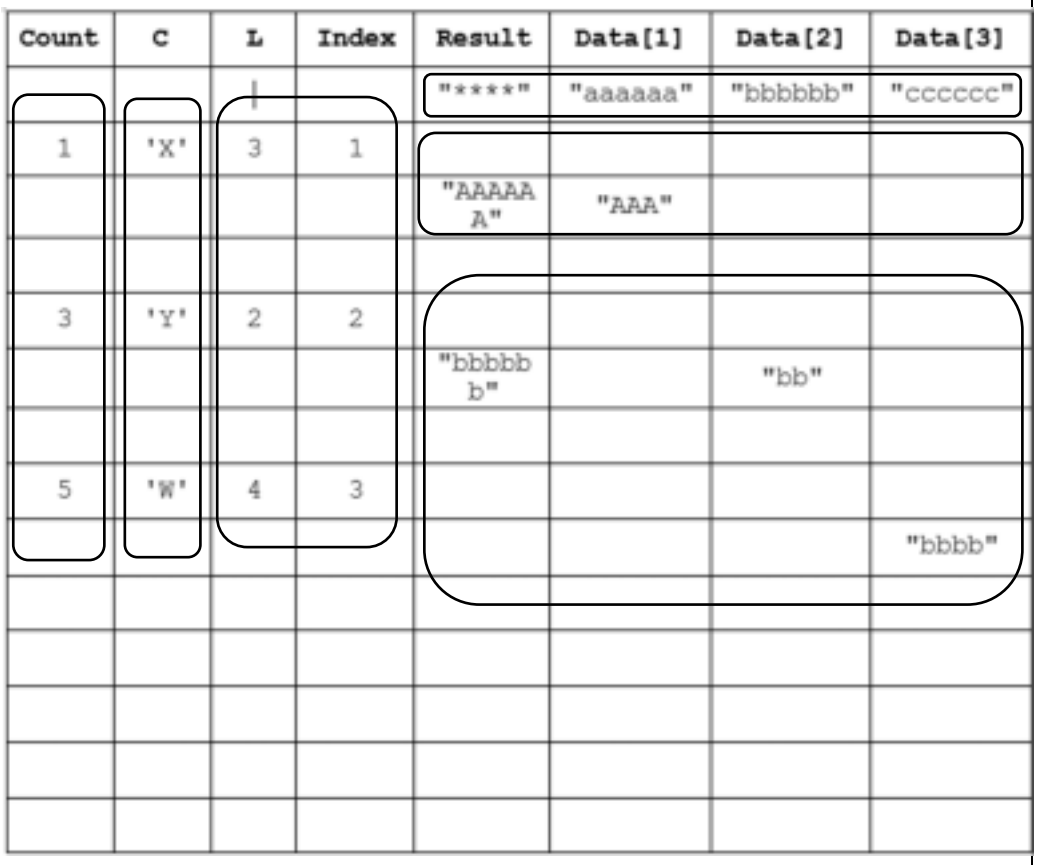
1 The PS contains a null pointer
2 The PF points to the first element on the free list
3 All the nodes are on the free list</td><td>3</td></tr>
<tr><td>3(b)</td><td>Max 2 marks for 'Variables':

<table><tr><td>The two variables will be of type Integer</td></tr><tr><td>The two variables will be used as pointers / indexes to the arrays.</td></tr><tr><td>The values stored in the two variables will indicate the first element in each list</td></tr><tr><td>The first 1D array will be of type String</td></tr><tr><td>The first 1D array will be used to store the values // data items // User IDs</td></tr><tr><td>The second 1D array will be of type Integer</td></tr><tr><td>The second 1D array will be used to store the pointers // point to next item</td></tr></table>
Mark as follows:

One mark for each of the first three rows
One mark for both Array 1 rows
One mark for both Array 2 rows</td><td>5</td></tr>
</table>

Question	Answer	Marks

4	Example:
	
	`FUNCTION Check() RETURNS STRING`
	`  DECLARE Odd, Even, Index : INTEGER`
	
	`  Odd ← 0`
	`  Even ← 0`
	
	`  FOR Index ← 1 TO 100`
	`  IF Index MOD 2 = 0 THEN`
	`  Even ← Even + Data[Index]`
	`  ELSE`
	`  Odd ← Odd + Data[Index]`
	`  ENDIF`
	`  NEXT Index`
	
	`ENDFUNCTION`
	
	Mark as follows:
	1. Function heading, ending and return type
	2. Declare local variables `Odd, Even` and `Index` as integers
	3. Initialise `Odd` and `Even`
	4. Loop for 100 `iterations` // through array
	5. Sum `Odd` and `Even` element values **in a loop**
	6. Compare `Odd` and `Even` after the loop and Return appropriate string
		6

Question	Answer	Marks
5(a)		**6**
	One mark per zone.	
5(b)(i)	OTHERWISE : CALL Error(C)	**1**
5(b)(ii)	After the 'Z' clause in the CASE construct // before the ENDCASE	**1**

Question	Answer	Marks

6(a)	`FUNCTION IsRA(x1, y1, x2, y2, x3, y3 : INTEGER) RETURNS BOOLEAN`
	`  DECLARE Len1, Len2, Len3 : INTEGER`
	
	`  Len1 ← (x1 - x2) ^ 2 + (y1 - y2) ^ 2`
	`  Len2 ← (x1 - x3) ^ 2 + (y1 - y3) ^ 2`
	`  Len3 ← (x2 - x3) ^ 2 + (y2 - y3) ^ 2`
	
	`  IF (Len1 = Len2 + Len3) OR__`
	`  (Len2 = Len1 + Len3) OR__`
	`  (Len3 = Len1 + Len2) THEN`
	`  RETURN TRUE`
	`  ELSE`
	`  RETURN FALSE`
	`  ENDIF`
	
	`ENDFUNCTION`
	
	Mark as follows:
	
	1 Calculate the square of the length of at least one side
	2 Calculation of all three lengths squared correctly
	3 **One** correct comparison of square of three lengths
	4 **All three** comparisons…
	5 combined using logical operators / nested `IF` // completely correct selection
	6 Return result correctly in both cases
		6

6(b)	1 mark for statement of problem: Problem: • The function will return an incorrect value // the test will fail 1 mark for solution: Solution: • Round the calculated values (to a known number of decimal places) • Define a threshold below which any difference can be ignored	**2**

Question	Answer	Marks
7(a)(i)	To make the solution easier to design / implement / solve	**1**

Question	Answer	Marks
7(a)(ii)	One mark per item **and** justification Item: mobile phone number Justification: to send the text message Item: name Justification: to personalise the text message Item: exercise interest Justification: to determine whether this member would be interested	**3**
7(a)(iii)	Examples include: • Add a member to a list of those interested in the new class • Remove the member from future SMS messages • Read/process Message • Identify who from One mark for each. **Max 2 marks**	**2**
7(b)(i)	Means that `Update` calls (one of) either `Sub-A, Sub-B` or `Sub-C` One mark for each point: • reference to selection / decision / if • naming all four modules correctly	**2**

7(b)(ii)	`PROCEDURE Sub-A (Name : STRING, BYREF P2 : BOOLEAN)`
	
	`FUNCTION Sub-B (P1 : REAL) RETURNS REAL`
	
	One mark per underlined part in each case
		4

Question	Answer	Marks

8(a)	`FUNCTION DeleteComment(Line : STRING) RETURNS STRING`
	`  DECLARE NewLine, TwoChars : STRING`
	`  DECLARE Count, TrimTo : INTEGER`
	`  CONSTANT Comment = "//"`
	
	`  NewLine ← Line`
	`  TrimTo ← 0`
	`  Count ← 1`
	`  WHILE Count < LENGTH(Line) AND TrimTo = 0`
	`  TwoChars ← MID(Line, Count, 2) // extract 2 chars`
	`  IF TwoChars = Comment THEN`
	`  TrimTo ← Count`
	`  ENDIF`
	`  Count ← Count + 1`
	`  ENDWHILE`
	
	`  IF TrimTo <> 0 THEN`
	`  NewLine ← LEFT(Line, TrimTo - 1)`
	`  ENDIF`
	
	`  RETURN NewLine`
	
	`ENDFUNCTION`
	
	Mark as follows:
	
	1 Loop to length of `Line` (parameter) // length of `Line -1`
	2 Terminate loop on first double slash
	3 Attempt to extract one/two characters **in a loop**
	4 Check for "//" after attempt at extraction **in a loop**
	5 Record start position of comment // Calculate amount of trim required
	6 Attempt to trim `Line` from start of comment
	7 Completely correct trimmed `Line` from start of comment
	8 Return `Line` after a reasonable overall attempt
		8

Question	Answer	Marks

8(b)	Example:
	
	`FUNCTION Stage_1(StudentName : STRING) RETURNS INTEGER`
	
	`DECLARE OldFile, NewFile, Line : STRING`
	`DECLARE Count : INTEGER`
	
	`OldFile ← StudentName & "_src.txt"`
	`NewFile ← StudentName & "_S1.txt"`
	`OPENFILE OldFile FOR READ`
	`OPENFILE NewFile FOR WRITE`
	`Count ← 0`
	
	`WHILE NOT EOF(OldFile)`
	`READFILE OldFile, Line`
	`Line ← DeleteComment(Line)`
	`IF LENGTH(Line) <> 0 THEN`
	`WRITEFILE NewFile, Line`
	`Count ← Count + 1`
	`ENDIF`
	`ENDWHILE`
	
	`CLOSEFILE OldFile`
	`CLOSEFILE NewFile`
	
	`RETURN Count`
	`ENDFUNCTION`
	
	Mark as follows:
	
	1 Generate filenames condone missing "_"
	2 Open both files in correct modes **and** subsequently close
	3 Loop to `EOF(OldFile)`
	4 Read a line from `OldFile` **and** execute `DeleteComment()` **in a loop**
	5 Skip blank lines after `DeleteComment()` **in a loop**
	6 Write `Line` to stage 1 file **and** increment Count
	7 Return the number of lines after a reasonable attempt at counting
		7