Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/23**

Paper 23 Problem Solving & Programming **May/June 2022**

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 2022 series for most Cambridge IGCSE, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

© UCLES 2022

[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 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.

© UCLES 2022

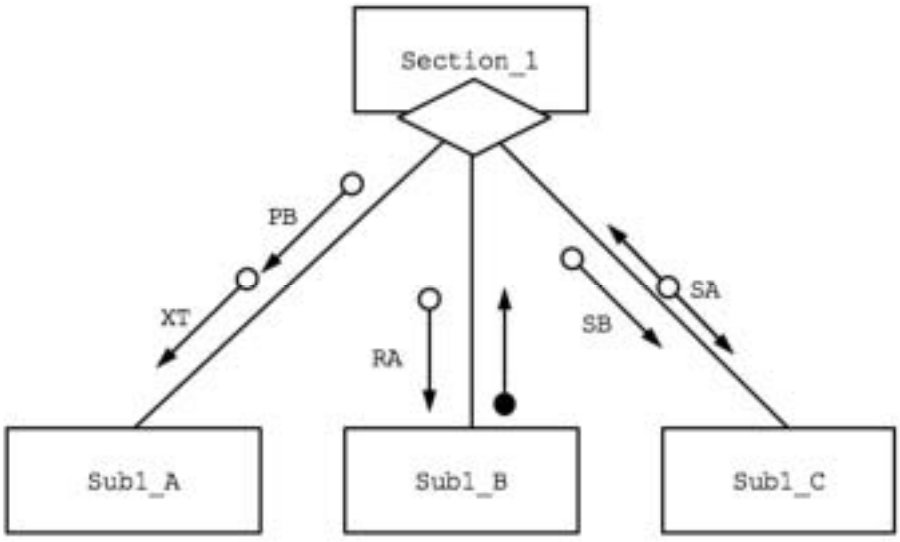
<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>1(a)</td><td>One mark per row

<table>
<tr><th>Pseudocode example</th><th>Selection</th><th>Iteration</th><th>Assignment</th></tr>
<tr><td><pre>FOR Index ← 1 TO 3
   Safe[Index] ← GetResult()
NEXT Index</pre></td><td></td><td>✓</td><td>✓</td></tr>
<tr><td><pre>OTHERWISE : OUTPUT "ERROR 1202"</pre></td><td>✓</td><td></td><td></td></tr>
<tr><td><pre>REPEAT UNTIL Index = 27</pre></td><td></td><td>✓</td><td></td></tr>
<tr><td><pre>INPUT MyName</pre></td><td></td><td></td><td>✓</td></tr>
<tr><td><pre>IF Mark > 74 THEN
   Grade ← 'A'
ENDIF</pre></td><td>✓</td><td></td><td>✓</td></tr>
</table></td><td>5</td></tr>
<tr><td>1(b)(i)</td><td>1 mark for any two rows correct
1 mark for all rows correct

<table>
<tr><th>Expression</th><th>Evaluation</th></tr>
<tr><td><pre>AAA AND (Count > 99)</pre></td><td>FALSE</td></tr>
<tr><td><pre>AAA AND (NOT BBB)</pre></td><td>TRUE</td></tr>
<tr><td><pre>(Count <= 99) AND (AAA OR BBB)</pre></td><td>TRUE</td></tr>
<tr><td><pre>(BBB AND Count > 50) OR NOT AAA</pre></td><td>FALSE</td></tr>
</table></td><td>2</td></tr>
<tr><td>1(b)(ii)</td><td>One mark from:

• To terminate a (conditional) loop when a value has been found
• When the variable can take <u>only one of two</u> possible values
• (Accept by example): When a variable is recording when an action has been done e.g. Yes or No // light is on</td><td>1</td></tr>
</table>

Question	Answer	Marks
2	Dependent marks One mark for type; one mark for matching reason Answers include: **Type**: Adaptive **Reason**: To accommodate a change in the requirement / technology / legislation e.g. a new HTML version is available **Type**: Corrective **Reason**: The program does not operate as expected / contains a bug e.g. passwords are not hidden	**4**

Question	Answer	Marks
3(a)	One mark per point to show: • module relationships / hierarchy / selection / repetition // how a problem is broken down • the parameters that are passed between the sub-tasks / modules // whether a module is a function or a procedure	**2**
3(b)	 One mark for each: 1 Four boxes linked as shown correctly labelled (Order of lower 3 **not** important) 2 Parameters to Sub1_A 3 Parameter to Sub1_B and return value 4 Parameters to Sub1_C 5 Selection diamond	**5**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>4</td><td>One mark per row 2 to 7 to Max 4
<table>
<tr><th>Type of test data</th><th>Example test value</th><th>Expected return value</th><th>Explanation</th></tr>
<tr><td>Normal</td><td>153</td><td>"PASS"</td><td>Value within the acceptable range</td></tr>
<tr><td>Abnormal</td><td>< 149</td><td>"FAIL"</td><td>Outside acceptable range / too small</td></tr>
<tr><td>Abnormal / Boundary</td><td>149</td><td>"FAIL"</td><td>Maximum unacceptable</td></tr>
<tr><td>Boundary / Extreme / Normal</td><td>150</td><td>"PASS"</td><td>Minimum acceptable</td></tr>
<tr><td>Boundary / Extreme / Normal</td><td>155</td><td>"PASS"</td><td>Maximum acceptable</td></tr>
<tr><td>Abnormal / Boundary</td><td>156</td><td>"FAIL"</td><td>Minimum unacceptable</td></tr>
<tr><td>Abnormal</td><td>>156</td><td>"FAIL"</td><td>Outside acceptable range / too large</td></tr>
</table></td><td>4</td></tr>
</table>

Question	Answer	Marks

5(a)(i)	`TYPE Employee`
	`  DECLARE EmployeeNumber : INTEGER`
	`  DECLARE Name : STRING`
	`  DECLARE Department : STRING`
	`  DECLARE Born : DATE`
	`  DECLARE Attendance : REAL`
	`ENDTYPE`
	
	One mark for each:
	
	1. `TYPE Employee` **and** `ENDTYPE`
	2. `Fields: EmployeeNumber` **and** `Name` **and** `Department`
	3. `Field: Born`
	4. `Field: Attendance`
		4

5(a)(ii)	`DECLARE Staff : ARRAY [1:500] OF Employee` One mark per underlined phrase	**2**

Question	Answer	Marks
5(b)(i)	Example answers to **Max 1**: • So that unused elements may be recognised when processing / searching • Otherwise the element may contain old / unexpected data	**1**
5(b)(ii)	Use of any 'impossible' field value, for example: • An `EmployeeNumber` field. e.g. < 1 • An empty string / impossible string e.g. "EMPTY" for name or department • DOB a long time ago... • Zero / Negative value for attendance	**1**

5(c)	`PROCEDURE Absentees()`
	`  DECLARE Index : INTEGER`
	`  FOR Index ← 1 TO 500`
	`  IF Staff[Index].EmployeeNumber <> -1 THEN  // not empty`
	`  IF Staff[Index].Attendance <= 90 THEN`
	`  OUTPUT Staff[Index].EmployeeNumber`
	`  OUTPUT Staff[Index].Name`
	`  ENDIF`
	`  ENDIF`
	`  NEXT Index`
	`ENDPROCEDURE`
	
	Marks as follows to **Max 4**:
	
	1 Procedure heading and ending **and** declaration of loop counter
	2 loop through 500 elements
	3 attempt to skip unused element
	4 test `Staff[Index].Attendance <= 90` **in a loop**
	5 if so, output `EmployeeNumber` and `Name` fields **in a loop**
		4

Question	Answer	Marks

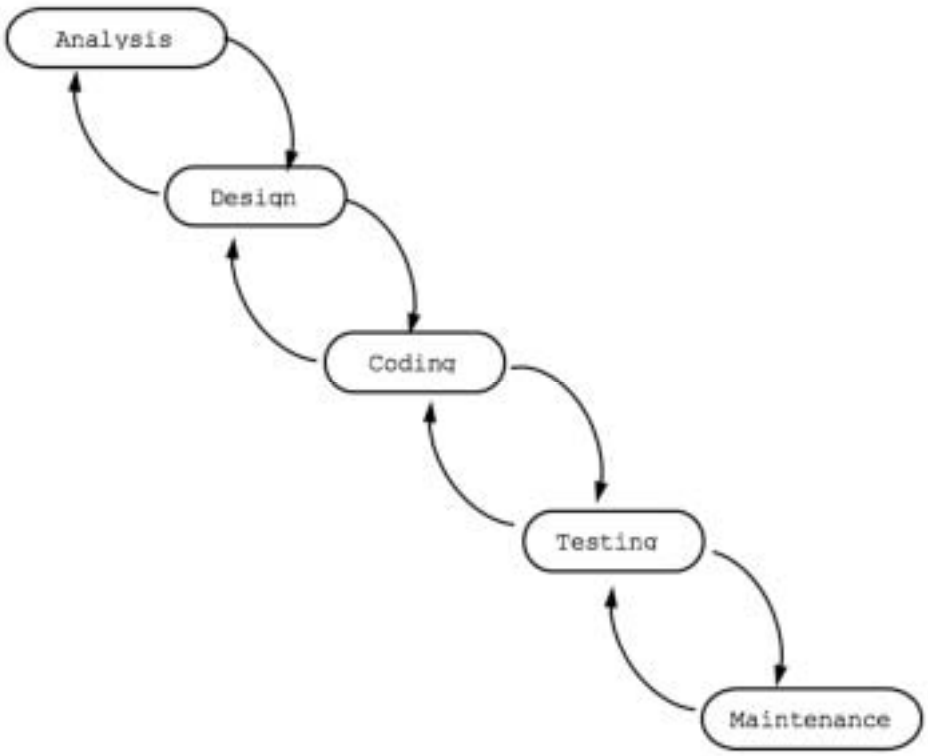
6(a)	`FUNCTION Factorial(ThisNum : INTEGER) RETURNS INTEGER`
	`  DECLARE Value : INTEGER`
	
	`  IF ThisNum < 0 THEN`
	`  Value ← -1`
	`  ELSE`
	`  Value ← 1`
	`  WHILE ThisNum <> 0`
	`  Value ← Value * ThisNum`
	`  ThisNum ← ThisNum – 1`
	`  ENDWHILE`
	`  ENDIF`
	
	`  RETURN Value`
	
	`ENDFUNCTION`
	
	Marks as follows to **Max 6**:
	
	1 Function heading and ending **including** parameter and return value
	2 Declaration and initialisation (to 1) of local Integer `Value` for result
	3 Check for illegal value (< 0)
	4 (Conditional) loop while `ThisNum` not zero // loop for `ThisNum` iterations
	5 Attempt to form answer by successive multiplication
	6 Completely correct MP5
	7 Return `INTEGER` value correctly in both cases: `ThisNum` < 0 and >= 0
	
	ALTERNATIVE RECURSIVE SOLUTION:
	
	`FUNCTION Factorial(ThisNum : INTEGER) RETURNS INTEGER`
	
	`  IF ThisNum > 1 THEN`
	`  RETURN ThisNum * Factorial(ThisNum – 1)`
	`  ELSE`
	`  IF ThisNum = 1 OR ThisNum = 0 THEN`
	`  RETURN 1`
	`  ELSE`
	`  RETURN -1`
	`  ENDIF`
	`  ENDIF`
	`ENDFUNCTION`
	
	Marks as follows:
	
	1 Function heading and ending **including** parameter and return value
	2 Test for `ThisNum > 1`
	3 and if so return product of `Thisnum` and `Factorial(ThisNum-1)`
	4 Check for special case...
	5 ...return 1 for 0 and 1 and return −1 for negative values
	6 Return `INTEGER` value (correctly in all cases)
		6

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>6(b)</td><td>One mark for:
• Rows A, B AND C
• Each of rows D, E and F

<table>
<tr><th>Label</th><th>Text</th></tr>
<tr><td>A</td><td>Is <code>Num</code> > 9?</td></tr>
<tr><td>B</td><td>YES</td></tr>
<tr><td>C</td><td>NO</td></tr>
<tr><td>D</td><td>Set <code><Identifier></code> to <code>Factorial(Num)</code></td></tr>
<tr><td>E</td><td>OUTPUT "Factorial of ", <code>Num</code>, " is ", <code><Identifier></code></td></tr>
<tr><td>F</td><td>Set <code>Num</code> to <code>Num + 1</code> // Increment <code>Num</code></td></tr>
</table></td><td>4</td></tr>
</table>

Question	Answer	Marks
7(a)(i)	The lines are output in an incorrect sequence / in the wrong order / not as they appear in the file	**1**
7(a)(ii)	**Description of error: (Max 2 marks)** • If the final line of the file is not written into array element 3 • then outputting the elements in the sequence 1 to 3 will give the error. **Explanation of error correction: (Max 2 marks)** • Attempt at description of 'shuffle' • Copy Buffer[2] to Buffer[1] **AND** copy Buffer[3] to Buffer[2] • Read a line from the file and write it to Buffer[3] **OR** • Store the index of the last element written to buffer (the last line of the file) • Replace the FOR loop with something that starts at index • and then wraps around (MOD 3) **OR (two-loop solution, not using an array)** • Loop to read file to end to get number of lines • close and re-open file • read (and discard) lines to number of lines - 3, then loop to read and output last 3 lines	**4**
7(b)	If the number of lines in the text file is a multiple of three	**1**

Question	Answer	Marks
7(c)	Correct answers include: `LineNum ← (LineNum MOD 3) + 1 // ((LineNum + 3) MOD 3) + 1` One mark for assignment to `LineNum` making any use of `MOD` One for completely correct statement	**2**

Question	Answer	Marks
8(a)	 One mark for: 1 (at least) three intermediate shapes with at least one valid stage names 2 Up and Down arrows between each pair of shapes 3 Design – Coding – Testing labels in correct sequence	**3**
8(b)	Downward arrows: result from one stage is input / passed to the next Upward arrows: more work is required at a previous stage to complete the current stage	**2**
8(c)	For example: Iterative / Rapid Application Development (RAD)	**1**

Question	Answer	Marks

9(a)	`FUNCTION Generate() RETURNS STRING`
	`  DECLARE Password, Group : STRING`
	`  DECLARE NextChar : CHAR`
	`  DECLARE ACount, BCount : INTEGER`
	`  CONSTANT HYPHEN = '-'`
	
	`  Password ← ""`
	
	`  FOR ACount ← 1 TO 3`
	`  Group ← ""`
	`  FOR BCount ← 1 TO 4`
	`  REPEAT`
	`  NextChar ← RandomChar()`
	`  UNTIL Exists(Group, NextChar) = FALSE`
	`  Group ← Group & NextChar`
	`  NEXT BCount`
	`  Password ← Password & Group & HYPHEN`
	`  NEXT ACount`
	`  Password ← LEFT(Password, 14)  // remove final hyphen`
	`  RETURN Password`
	`ENDFUNCTION`
	
	Marks as follows to **Max 7**:
	
	1 Declaration **and** initialisation of `Password` as `STRING`
	2 Outer loop for three groups / until password is complete // three group loops
	3 Attempt to use of **both** `RandomChar()` **and** `Exists()` **in a loop**
	4 (Inner) loop for 4 characters in a group // note every 4 chars **in a loop**
	5 Conditional loop until char is unique
	6 Concatenating unique character to `Group` **in a loop**
	7 Concatenate `Group` / random character to `Password` **in a loop**
	8 (Attempt to) insert hyphens between groups (or removing later) **and** Return `Password`
		7

Question	Answer	Marks

9(b)	`FUNCTION AddPassword(Name, Password : STRING)`
	`  RETURNS BOOLEAN`
	`  DECLARE Index : INTEGER`
	`  DECLARE Added : BOOLEAN`
	
	`  Added ← FALSE`
	`  Index ← 1`
	
	`  IF FindPassword(Name) = "" THEN // Domain name not in`
	`  // array`
	`  WHILE Added = FALSE AND Index <= 500`
	`  IF Secret[Index, 1] = "" THEN`
	`  Secret[Index, 1] ← Name`
	`  Secret[Index, 2] ← Encrypt(Password)`
	`  Added ← TRUE`
	`  ELSE`
	`  Index ← Index + 1`
	`  ENDIF`
	`  ENDWHILE`
	`  ENDIF`
	
	`  RETURN Added`
	
	`ENDFUNCTION`
	
	Marks as follows:
	
	1 Check that the website domain name isn't already in array using `FindPassword()` / linear search, otherwise:
	2 (Conditional) loop while not added and not end of array
	3 Check for unused element by testing value in column 1 **in a loop**
	4 If unused, write parameter values to column 1 and 2 **and** set flag / variable
	5 ...having used `Encrypt()` on the password
	6 Return `BOOLEAN` value (correctly in all cases)
		6

9(c)	One mark per point to **Max 3**. **Solution based on field length:** • Convert the length of the website domain name (either field) … • … to a string of fixed length • Form a string by concatenate this string with the other two (and write as one line of the file) **Solution based on use of separator character:** • Select a (separator) character that cannot occur in the domain name (e.g. space) • Create a string from the domain name followed by the separator • ...Concatenate the encrypted password (and write as one line of the file)	**3**