Cambridge Assessment International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2023**

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 2023 series for most Cambridge IGCSE, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

© UCLES 2023

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 2023

Question	Answer	Marks
1(a)(i)	Use of constants	**1**
1(a)(ii)	One mark per bullet point (or equivalent to max 3): 1 Postal rates are entered once only 2 Avoids input error / changing the cost accidentally // avoids different values for postal rates at different points in the program 3 When required, the constant representing the postal rate value is changed once only // easier to maintain the program when the postal rates change 4 Makes the program easier to understand **Note: Max 3 marks**	**3**
1(b)	One mark per bullet point: • Indentation • White space • Comments • Sensible / meaningful variable names // use of Camel Case • Capitalised keywords **Note: Max 3 marks**	**3**
1(c)	One mark per bullet point: • BOOLEAN • REAL • STRING	**3**

Question	Answer	Marks
2(a)	`MyDOB ← SETDATE(17, 11, 2007)`	**1**
2(b)	`NumMonths ← 12 - MONTH(MyDOB)` One mark per underlined part	**2**

Question	Answer	Marks
2(c)	One mark per array definition bullet: • A (1D) array containing 7 elements • of type STRING One mark per Step: Step1: Assign value "Sunday" to first element, "Monday" to second element etc. Step2: Use the function `DAYINDEX()` to return / find the day number from `MyDoB` Step3: Use the returned value as the array index / to access the element that contains the name / string Step4: Output the element / name / string **Note: Max 2 for Array definition, Max 4 for steps**	**6**

Question	Answer	Marks
3(a)(i)	One mark per point: 1 Check that the queue is not full 2 `EoQ` pointer will move to point to location 9 3 Data item Orange will be stored in location referenced by EoQ pointer 4 `EoQ` pointer will move to point to location 0 5 Data item Yellow will be stored in location referenced by EoQ pointer **Note: max 4 marks**	**4**
3(a)(ii)	7	**1**
3(b)	One mark per bullet: 1 Open file in `READ` mode 2 Loop to `EOF()` // read / process all the lines in file 3 Loop will end when return value from `AddToQueue()` is `FALSE` / queue is full 4 Read a line from the file in a loop 5 Pass string to `AddToQueue()` // `AddToQueue()` is executed with line as parameter	**5**

Question	Answer	Marks

4	`Function GetNum(ThisString : STRING, ThisChar : CHAR) RETURNS INTEGER`
	`  DECLARE Index, Count : INTEGER`
	
	`  Count ← 0`
	
	`  FOR Index ← 1 TO LENGTH(ThisString)`
	`  IF MID(ThisString, Index, 1) = ThisChar THEN`
	`  Count ← Count + 1`
	`  ENDIF`
	`  NEXT Index`
	`  RETURN Count`
	
	`ENDFUNCTION`
	
	Mark as follows:
	1 Function heading and end, including parameters and return type
	2 Declare local Integers for `Index` and `Count`
	3 Loop for length of `ThisString`
	4 Extract a character and compare with <u>parameter</u> **in a loop**
	5 Increment `Count` if match **in a loop**
	6 Return `Count` **after loop**
		6

Question	Answer	Marks
5(a)	One mark per point: • parameter / `Num` has been passed by reference // should have been passed by value • so when the value / `ThisNum` is modified (in procedure `DisplaySqrt()`) • the new value will be used in the loop (lines 40–43) // `Num` will be changed to modified value	**3**
5(b)	• The rules of the language have not been broken // there are no syntax errors	**1**
5(c)	Could use an IDE to: • Set a breakpoint to stop the program at a certain line / statement / point • Step through the program line by line / statement by statement • checking the value of `'num'` / a variable using a report / watch window One mark per bullet	**3**

Question	Answer	Marks
5(d)	Answers include: • Change the statement into a comment • Change the statement to a string representing a literal value and assign it to a variable / output it **Note: max 1 mark**	**1**

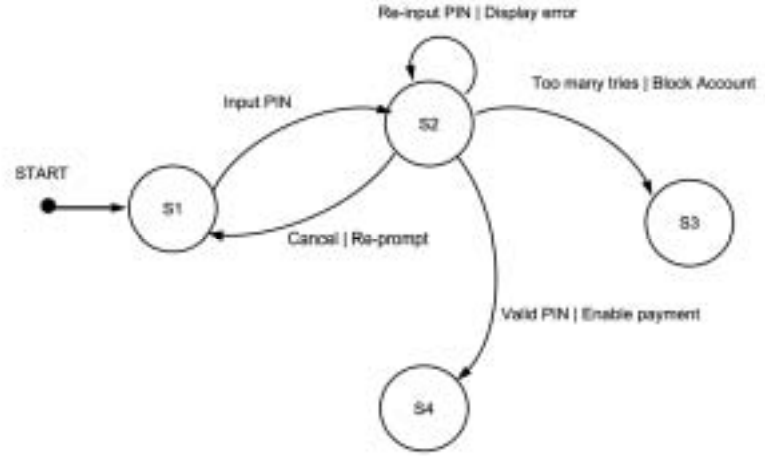
Question	Answer	Marks

6	Example of iterative solution:
	
	`PROCEDURE Square(Dim : INTEGER)`
	`  DECLARE Count : INTEGER`
	`  DECLARE ThisChar : CHAR`
	`  DECLARE StringA, StringB : STRING`
	`  CONSTANT FILLER = '*'`
	
	`  StringA ← ""`
	
	`  ThisChar ← NUM_TO_STR(Dim)`
	
	`  FOR Count ← 1 TO Dim`
	`  StringA ← StringA & ThisChar //build up first & last line`
	`  NEXT Count`
	
	`  StringB ← ThisChar`
	`  FOR Count ← 1 TO Dim - 2`
	`  StringB ← StringB & FILLER //build up intermediate line`
	`  NEXT Count`
	`  StringB ← StringB & ThisChar // add final digit`
	
	`  OUTPUT StringA`
	`  FOR Count ← 1 TO Dim - 2`
	`  OUTPUT StringB`
	`  NEXT Count`
	
	`  IF Dim <> 1 THEN`
	`  OUTPUT StringA`
	`  ENDIF`
	
	`ENDPROCEDURE`
	
	For loop-based solutions, mark as follows:
	1 Procedure heading and ending including parameter
	2 Loop using parameter, containing attempt to construct first line / last line
	3 Construct first line / last line
	4 Attempt at loop to construct intermediate line
	5 Output first / last line of square when `Dim > 2`
	6 Output all intermediate lines **in a loop**
	7 Correct output of first two squares
	
	Note: Max 6 marks
		6

Question	Answer	Marks

6	Example of selection-based solution:
	
	`PROCEDURE Square(Dim : INTEGER)`
	`  DECLARE Count : INTEGER`
	
	`  CASE OF Dim`
	`  1 : OUTPUT "1"`
	`  2 : OUTPUT "22"`
	`  OUTPUT "22"`
	`  3 : OUTPUT "333"`
	`  OUTPUT "3*3"`
	`  OUTPUT "333"`
	`  4 : OUTPUT "4444"`
	`  FOR Count ← 1 TO 2`
	`  OUTPUT "4**4"`
	`  NEXT Count`
	`  OUTPUT "4444"`
	`  5 : OUTPUT "55555"`
	`  FOR Count ← 1 TO 3`
	`  OUTPUT "5***5"`
	`  NEXT Count`
	`  OUTPUT "55555"`
	`  6 : OUTPUT "666666"`
	`  FOR Count ← 1 TO 4`
	`  OUTPUT "6****6"`
	`  NEXT Count`
	`  OUTPUT "666666"`
	`  7 : OUTPUT "7777777"`
	`  FOR Count ← 1 TO 5`
	`  OUTPUT "7*****7"`
	`  NEXT Count`
	`  OUTPUT "7777777"`
	`  8 : OUTPUT "88888888"`
	`  FOR Count ← 1 TO 6`
	`  OUTPUT "8******8"`
	`  NEXT Count`
	`  OUTPUT "88888888"`
	`  9 : OUTPUT "999999999"`
	`  FOR Count ← 1 TO 7`
	`  OUTPUT "9*******9"`
	`  NEXT Count`
	`  OUTPUT "999999999"`
	`  ENDCASE`
	
	`ENDPROCEDURE`
	
	For in-line / selection-based solutions, mark as follows:
	1 Procedure heading and ending including parameter
	2 Correct use of parameter to select all required squares
	3 Correct output of first two squares
	4 At least six squares correctly output
	5 At least one loop to output lines with multiple asterisks
	6 All number squares output correctly	

Question	Answer	Marks
7(a)(i)	Correct answers include: • Information: customer name Justification: to personalise / address the email • Information: email address Justification: so that the email can be delivered correctly • Information: product category preference Justification: to check that the customer would be interested in the product • Information: contact preference Justification: to check that the customer will accept contact via email • Information: order history Justification: to send details of product similar to that already bought // to identify frequent shoppers • Information: new product information Justification: to include information about the new product in the email One mark for each piece of information **and** matching justification **Note: Max 3 marks**	**3**
7(a)(ii)	One mark for each piece of information **and** matching justification: • postal address Justification: sending an email, not a letter • payment details Justification: Nothing being bought / sold at this stage • order history Justification: Customer preference used to decide if new product is relevant **Note: Max 2 marks**	**2**

Question	Answer	Marks
7(b)	 One mark for each: 1 Line from S1 to S2 **and** label 2 S2 loop label 3 S3 circle and label on incoming 4 S4 circle and label on incoming	**4**

Question	Answer	Marks

8(a)	`FUNCTION CheckInfo(NewLine: STRING) RETURNS BOOLEAN`
	`  DECLARE ThisNum : STRING`
	`  DECLARE Index : INTEGER`
	
	`  IF LENGTH(NewLine) < 19 THEN`
	`  RETURN FALSE`
	`  ENDIF`
	
	`  FOR Index ← 1 TO 4`
	`  IF NOT IS_NUM(MID(NewLine, Index, 1)) THEN`
	`  RETURN FALSE`
	`  ENDIF`
	`  NEXT Index`
	
	`  ThisNum ← LEFT(Newline, 4)`
	
	`  IF ThisNum < "0001" OR ThisNum > "5999" THEN`
	`  RETURN FALSE`
	`  ENDIF`
	
	`  IF NOT OnlyAlpha(MID(Newline, 5, 3)) THEN`
	`  RETURN FALSE`
	`  ENDIF`
	
	`  RETURN TRUE`
	
	`ENDFUNCTION`
	
	Mark as follows:
	1 Test length of parameter
	2 Extract first 4 characters of parameter (as `ItemNum`)
	3 Test first four characters are **all** numeric
	4 Test ItemNum in range "0001" to "5999"
	5 Extract characters 5 to 7 of parameter (as `SupplierCode`)
	6 Use of `OnlyAlpha()` with extracted `SupplierCode`
	7 Return `BOOLEAN` value correctly in all cases, must have been declared as local
		7

Question	Answer	Marks

8(b)	`PROCEDURE AddItem(NewLine : STRING)`
	`  DECLARE NewItemNum, ThisItemNum : STRING`
	
	`  OPENFILE "Stock.txt" FOR READ`
	`  OPENFILE "NewStock.txt" FOR WRITE`
	`  NewItemNum ← LEFT(NewLine, 4)`
	
	`  WHILE NOT EOF("Stock.txt")`
	`  READFILE("Stock.txt", ThisLine)`
	`  ThisItemNum ← LEFT(ThisLine, 4)`
	`  IF ThisItemNum > NewItemNum THEN`
	`  WRITEFILE("NewStock.txt", NewLine) // write New Line...`
	`  NewItemNum ← "9999" // ...once only`
	`  ENDIF`
	`  WRITEFILE("NewStock.txt", ThisLine)`
	`  ENDWHILE`
	
	`  IF NewItemNum <> "9999" THEN`
	`  WRITEFILE("NewStock.txt", NewLine) //New last line in the file`
	`  ENDIF`
	
	`  CLOSEFILE "Stock.txt"`
	`  CLOSEFILE "NewStock.txt"`
	`ENDPROCEDURE`
	
	Mark as follows:
	1 Open both files, in correct modes, and subsequently close
	2 Conditional loop until end of file `Stock.txt`
	3 Read a line from `Stock.txt` **AND** extract `ThisItemNum` **in a loop**
	4 Test `ThisItemNum > NewItemNum` then write `NewLine` to `NewStock.txt`
	5 ...including mechanism to only do this **once only**
	6 Write line read from `Stock` to `NewStock`
	7 Deal with the case where `NewLine` is the new last line
		7

Question	Answer	Marks

8(b)	Example of array-based solution:
	
	`PROCEDURE AddItem(NewLine : STRING)`
	`  DECLARE ThisItemNum, ThisLine : STRING`
	`  DECLARE Temp : ARRAY [1:5999] OF STRING`
	`  DECLARE Index : INTEGER`
	
	`  FOR Index ← 1 TO 5999`
	`  Temp[Index] ← "" //Initialise array`
	`  NEXT Index`
	
	`  Index ← STR_TO_NUM(LEFT(NewLine, 4))`
	`  Temp[Index] ← NewLine  //Add new line to array`
	
	`  OPENFILE "Stock.txt" FOR READ`
	
	`  WHILE NOT EOF("Stock.txt")`
	`  READFILE("Stock.txt", ThisLine)`
	`  Index ← STR_TO_NUM(LEFT(ThisLine, 4))`
	`  Temp[Index] ← ThisLine //Add line from file to array`
	`  ENDWHILE`
	`  CLOSEFILE "Stock.txt"`
	
	`  OPENFILE "NewStock.txt" FOR WRITE`
	`  FOR Index ← 1 TO 5999`
	`  IF Temp[Index] <> "" THEN  //Write non-blank element...`
	`  WRITEFILE("NewStock.txt", Temp[Index]) //...to new file`
	`  ENDIF`
	`  NEXT Index`
	`  CLOSEFILE "NewStock.txt"`
	
	`ENDPROCEDURE`
	
	Mark as follows:
	1 Open both files, in correct modes, and subsequently close
	2 Declare **AND** initialise `Temp` array
	3 Store `NewLine` in appropriate array element
	4 Loop until end of file `Stock.txt`
	5 Read a line from `Stock.txt` **AND** extract `Index` **in a loop**
	6 Assign line read to appropriate array element **in a loop**
	7 Loop through array, writing non-blank elements to file `NewStock.txt`	

Question	Answer	Marks
8(c)	One mark for method: Method: Stub testing Two marks for description: • The modules `SuppExists()` and `CheckSupplier()` are replaced by dummy modules • ...which return a known result / contain an output statement to show they have been called	**3**