



Cambridge International AS & A Level

COMPUTER SCIENCE

9618/23

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 **11** 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)(i)	One from: - The program obeys the rules / grammar of the programming language used - The program will run // it can be compiled / interpreted - Accept by example. e.g. 'no mis-spelt keywords' / 'all brackets match'	1										
1(a)(ii)	One mark for type plus one for corresponding description Type of error: A logic error Description: - An error in the algorithm / design of the solution - the program does not behave as expected / give the expected output. - Accept by example e.g. wrong arithmetic operator used / wrong loop count OR Type of error: Run-time error Description: - The program performs an illegal instruction / invalid operation - Accept by example: divide by zero or endless loop or simply 'crashes' / freezes	2										
1(b)	<table><tr><th>Use of variable</th><th>Data type</th></tr><tr><td>The average mark in a class of 40 students</td><td>REAL</td></tr><tr><td>An email address</td><td>STRING</td></tr><tr><td>The number of students in the class</td><td>INTEGER</td></tr><tr><td>Indicate whether an email has been read</td><td>BOOLEAN</td></tr></table> One mark per row	Use of variable	Data type	The average mark in a class of 40 students	REAL	An email address	STRING	The number of students in the class	INTEGER	Indicate whether an email has been read	BOOLEAN	4
Use of variable	Data type											
The average mark in a class of 40 students	REAL											
An email address	STRING											
The number of students in the class	INTEGER											
Indicate whether an email has been read	BOOLEAN											

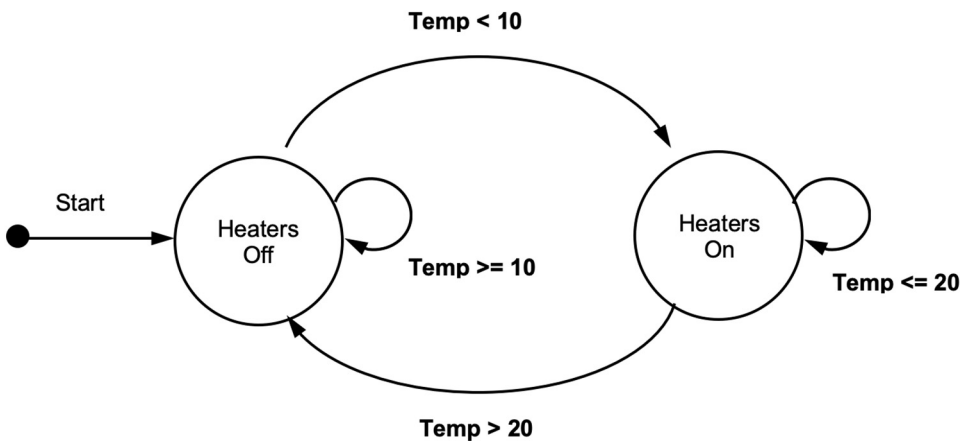
Question	Answer	Marks																					
1(c)(i)	<table border="1"> <thead> <tr> <th>Information</th><th>Essential</th><th>Not essential</th></tr> </thead> <tbody> <tr> <td>Departure time</td><td>✓</td><td></td></tr> <tr> <td>Flight Number</td><td></td><td>✓</td></tr> <tr> <td>Departure airport</td><td>✓</td><td></td></tr> <tr> <td>Aircraft type</td><td></td><td>✓</td></tr> <tr> <td>Ticket price</td><td>✓</td><td></td></tr> <tr> <td>Number of seats in aircraft</td><td></td><td>✓</td></tr> </tbody> </table> One mark for two rows correct Two mark for four rows correct Three mark for all rows correct	Information	Essential	Not essential	Departure time	✓		Flight Number		✓	Departure airport	✓		Aircraft type		✓	Ticket price	✓		Number of seats in aircraft		✓	3
Information	Essential	Not essential																					
Departure time	✓																						
Flight Number		✓																					
Departure airport	✓																						
Aircraft type		✓																					
Ticket price	✓																						
Number of seats in aircraft		✓																					
1(c)(ii)	One mark for technique and one for benefit, Max 1 mark for 'Benefit' Technique: Abstraction Benefit: - The solution is simplified so easier / quicker to design / implement - The system is tailored to the need of the user	2																					
1(c)(iii)	Answers include: - Destination / arrival airport - Arrival time / flight duration - Date of flight - Seat number - Seat availability Max 2 marks	2																					

Question	Answer	Marks
2(a)	One mark for reference to: - The use a variable as an index to the array - A loop to iterate through the array - An Inner loop (with a reducing range) - Test if current element is greater than next element - if so then swap elements - Description of swap - Attempt at efficient algorithm Max 6 marks	6

Question	Answer	Marks
2(b)	<pre> Count ← 1 Flag ← FALSE WHILE Flag = FALSE AND Count ≤ 5 CALL ReBoot() Count ← Count + 1 Flag ← Check() ENDWHILE IF Flag = FALSE THEN CALL Alert(27) ENDIF </pre> One mark per point: 1 Initialisation of Count AND Flag 2 WHILE ... ENDWHILE // REPEAT ... UNTIL loop 3 ... including both conditions 4 Call ReBoot() AND increment Count inside the loop 5 Assign return value from Check() to Flag inside the loop 6 Final test of Flag AND call to Alert(27) not in a loop	6

Question	Answer	Marks									
3(a)(i)	One mark per point: • EoQ pointer will move to point to location 4 // incremented EoQ (by 1) • Data value "Octopus" will be stored in location pointed to be EoQ / location 4	2									
3(a)(ii)	One mark for each bullet • Value "Frog" // value pointed to by FoQ / location 0 is assigned to variable AnimalName • FoQ pointer will move to point to location 1 / point to "Cat" // incremented FoQ (by 1) <table><tr><td>0</td><td>Frog</td><td rowspan="4">← Front of queue pointer</td></tr><tr><td>1</td><td>Cat</td></tr><tr><td>2</td><td>Fish</td></tr><tr><td>3</td><td>Elk</td></tr></table> ← End of queue pointer	0	Frog	← Front of queue pointer	1	Cat	2	Fish	3	Elk	2
0	Frog	← Front of queue pointer									
1	Cat										
2	Fish										
3	Elk										
3(a)(iii)	There is only one data item in the queue	1									

Question	Answer	Marks																		
3(b)(i)	One mark for data values plus one mark for pointers <table><tr><td>0</td><td>Frog</td><td rowspan="4">← Front of queue pointer</td></tr><tr><td>1</td><td>Cat</td></tr><tr><td>2</td><td>Fish</td></tr><tr><td>3</td><td>Elk</td></tr><tr><td>4</td><td>Wasp</td><td rowspan="4">← End of queue pointer</td></tr><tr><td>5</td><td>Bee</td></tr><tr><td>6</td><td>Mouse</td></tr><tr><td>7</td><td></td></tr></table> One mark for each pointer One mark for three new data values	0	Frog	← Front of queue pointer	1	Cat	2	Fish	3	Elk	4	Wasp	← End of queue pointer	5	Bee	6	Mouse	7		3
0	Frog	← Front of queue pointer																		
1	Cat																			
2	Fish																			
3	Elk																			
4	Wasp	← End of queue pointer																		
5	Bee																			
6	Mouse																			
7																				
3(b)(ii)	<table><tr><td>0</td><td>Shark</td><td rowspan="4">← End of queue pointer</td></tr><tr><td>1</td><td>(Cat)</td></tr><tr><td>2</td><td>(Fish)</td></tr><tr><td>3</td><td>(Elk)</td></tr><tr><td>4</td><td>Wasp</td><td rowspan="4">← Front of queue pointer</td></tr><tr><td>5</td><td>Bee</td></tr><tr><td>6</td><td>Mouse</td></tr><tr><td>7</td><td>Dolphin</td></tr></table> One mark for BOTH pointers One mark for all data values as shown	0	Shark	← End of queue pointer	1	(Cat)	2	(Fish)	3	(Elk)	4	Wasp	← Front of queue pointer	5	Bee	6	Mouse	7	Dolphin	2
0	Shark	← End of queue pointer																		
1	(Cat)																			
2	(Fish)																			
3	(Elk)																			
4	Wasp	← Front of queue pointer																		
5	Bee																			
6	Mouse																			
7	Dolphin																			
3(c)	One mark per point: 1 If incremented $E \circ Q = F \circ Q$ then error condition: queue is full 2 Increment the $E \circ Q$ 3 Manage wrap-around	3																		

Question	Answer				Marks
4(a)	Test	Test data value	Explanation	Expected Outcome	4
	1	23	Normal Data	Data is accepted	
	2	0	Boundary Data	Data is accepted	
	3	40	Boundary Data	Data is accepted	
	4	>= 41	Abnormal Data	Data is rejected	
	5	<= -1	Abnormal Data	Data is rejected	
	One mark per row for rows 2 to 5.				
4(b)	 One mark for each label: - Temp < 10 (Heaters Off to Heaters On) - Temp > 20 (Heaters On to Heaters Off) - on BOTH loops (non-contradictory values)				3

Question	Answer	Marks
5(a)	One mark for the character and one for the corresponding reason. - Character: Any except alphabetic, numeric, ',' ':' or space - Reason: character doesn't occur in data to be recorded	2
5(b)	Design	1

Question	Answer	Marks
5(c)	<pre> FUNCTION LogEvents(StudentID : STRING) RETURNS INTEGER DECLARE FileData : STRING DECLARE Index, Count : INTEGER CONSTANT LogFile = "LogFile" Count ← 0 OPENFILE LogFile FOR APPEND FOR Index ← 1 TO 2000 FileData ← LogArray[Index] IF LEFT(FileData, 6) = StudentID THEN WRITEFILE (LogFile, FileData) //brackets optional Count ← Count + 1 LogArray[Index] ← "" // clear the element ENDIF NEXT Index CLOSEFILE LogFile RETURN Count ENDFUNCTION </pre> 1 mark for each of the following: 1 Function heading and ending including parameter and return type 2 OPEN file LogFile for APPEND and subsequent CLOSE 3 Loop for 2000 iterations 4 Extract first 6 characters from array element in a loop 5 Compare first 6 characters with parameter in a loop 6 If equal: • write whole array element string to file and • increment Count and • clear array element in a loop 7 Return Count (must have been declared and initialised)	7

Question	Answer	Marks
6(a)	<pre> PROCEDURE SetRow(Row, SkipNum, SetNum : INTEGER) DECLARE Col : INTEGER // array is 1280 x 800 FOR Col ← SkipNum + 1 TO SkipNum + SetNum Screen[Row, Col] ← 1 NEXT Index ENDPROCEDURE ALTERNATIVE 1: FOR Col ← 1 TO SetNum Screen[Row, SkipNum + Col] ← 1 NEXT Col ALTERNATIVE 2: WHILE SetNum > 0 Screen[Row, SkipNum + SetNum] ← 1 SetNum ← SetNum - 1 ENDWHILE Mark as follows: 1 Procedure heading and ending including parameters 2 Declaration of local Integer for Col 3 Count-controlled loop with meaningful start number 4 correct stop number 5 Reference Screen Array element and set to 1 in a loop </pre>	5

Question	Answer	Marks
6(b)	<pre> FUNCTION SearchInRow(ThisRow, StartCol : INTEGER) RETURNS INTEGER DECLARE ThisCol, Step : INTEGER DECLARE Found: BOOLEAN // array is 1280 x 800 Found ← FALSE ThisCol ← StartCol // first decide which way to search IF StartCol = 1 THEN Step ← 1 EndCol ← 1281 ELSE Step ← -1 EndCol ← 0 ENDIF WHILE ThisCol <> EndCol AND Found = FALSE IF Screen[ThisRow, ThisCol] <> 1 THEN ThisCol ← ThisCol + Step ELSE Found ← TRUE ENDIF ENDWHILE IF Found = FALSE THEN ThisCol ← -1 ENDIF RETURN ThisCol ENDFUNCTION </pre> Mark as follows: 1 Interpreting <code>StartCol</code> parameter to determine direction of search 2 An attempt at searching both up and down 3 Conditional Loop / Count-controlled loop with use of <code>ThisCol</code> index 4 Using correct values for <code>StartCol</code>, <code>EndCol</code> and <code>Step</code> 5 Reference a <code>Screen</code> element and compare with 1 in a loop 6 If equal save column or immediately Return column in a loop 7 Return column number or -1 Loop(s) terminate when element with value = 1 found Max 7 marks if function heading, including return type, and ending is incorrect or incomplete	8

Question	Answer	Marks
6(c)	<pre> FUNCTION GetCentreCol(ThisRow : INTEGER) RETURNS INTEGER DECLARE StartCol, EndCol, CentreCol : INTEGER StartCol ← SearchInRow(ThisRow, 1) IF StartCol = -1 THEN CentreCol ← StartCol ELSE EndCol ← SearchInRow(ThisRow, 1280) CentreCol ← INT((StartCol + EndCol)/2) ENDIF RETURN CentreCol ENDFUNCTION </pre> Mark as follows: 1 Declaration of local INTEGER for return value 2 Use SearchInRow() with correct parameters and check for -1 3 Use SearchInRow(ThisRow, 1) and SearchInRow(ThisRow, 1280) 4 Calculate centre column 5 Use of INT() function // use of DIV 6 Return -1 or centre value Max 5 marks if function heading, including return type, and ending is incorrect or incomplete	6