CAMBRIDGE
International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/21**

Paper 2 Fundamental Problem-solving and Programming Skills **May/June 2026**

MARK SCHEME

Maximum Mark: 75

Published

This mark scheme contains the instructions and marking guidance that our examiners used to mark candidate responses for this component.

Before they start marking, examiners complete a standardisation process. They review a range of candidate responses and discuss the mark scheme in detail to agree which answers can and cannot be accepted. Examiners award marks where it is clear that candidate responses have met the requirements of the mark scheme. These requirements may vary between different components or different versions of the same component, depending on the exact requirements of the question and the candidate responses seen during the standardisation process.

Sometimes, our mark schemes may allow marks to be awarded to responses which contain elements that are factually incorrect or for content not covered by the syllabus. We do this in response to the demands of a specific question to support candidates and make sure that our assessments are fair. However, these allowances should not be used as a guide for teaching and learning.

We cannot discuss the mark scheme with schools, and we recommend that schools read the mark scheme together with the assessment material and the Principal Examiner Report for Teachers.

This document consists of **14** printed pages.

[Turn over

General marking principles

All examiners must apply these marking principles when marking candidate responses.

1	Examiners must award marks for each response in line with: • the specific **content** defined in the mark scheme • the specific **skills** defined in the mark scheme or in the level descriptions • the **standard of response** required as exemplified by the standardisation scripts.
2	Examiners must award **whole marks** (not half marks, or other fractions).
3	Examiners must award marks **positively**. This means that: • marks are not deducted for errors or omissions • marks are awarded for correct/valid answers as defined in the mark scheme and also for **other valid answers** which go beyond the scope of the syllabus and mark scheme referring to your team leader as appropriate • the quality of spelling, punctuation and grammar are judged **only** when the mark scheme indicates that these features are **specifically assessed** in the question. However, the meaning of all responses must be unambiguous.
4	Examiners must **consistently** apply the requirements of the mark scheme and any rules for what to do when candidates have not followed instructions correctly.
5	Examiners must use the **full range of marks** defined in the mark scheme, unless limited by the quality of the candidate responses seen.
6	Examiners must award marks according to the mark scheme and **must not** award marks with grade thresholds or grade descriptions in mind.

Annotations guidance for centres

Examiners use a system of annotations as a shorthand for communicating their marking decisions to one another. Examiners are trained during the standardisation process on how and when to use annotations. The purpose of annotations is to inform the standardisation and monitoring processes and guide the supervising examiners when they are checking the work of examiners within their team. The meaning of annotations and how they are used is specific to each component and is understood by all examiners who mark the component.

We publish annotations in our mark schemes to help centres understand the annotations they may see on copies of scripts. Note that there may not be a direct correlation between the number of annotations on a script and the mark awarded. Similarly, the use of an annotation may not be an indication of the quality of the response.

The annotations listed below were available to examiners marking this component in this series.

Annotations

Annotation	Meaning
	Benefit of the doubt
	To indicate where a key word/phrase/code is missing
	Incorrect
	Follow through
	Indicate a point in an answer
Highlighted text	To draw attention to a particular aspect or to indicate where parts of an answer have been combined
	Ignore
	Not answered question
	No examples or not enough
	Not relevant or used to separate parts of an answer
Off-page comment	Allows comments to be entered at the bottom of the RM marking window and then displayed when the associated question item is navigated to.
	Repetition
	Indicates that work or a page has been seen including blank answer spaces and blank pages.
	Correct

Annotation	Meaning
TV	Too vague

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.

Question	Answer	Marks
1(a)	Count-control(led) loop // Count-control(led) iteration	1
1(b)	4	1

Question	Answer	Marks
2(a)(i)	BOOLEAN	1
2(a)(ii)	MainsOn (= TRUE) AND WaterOn (= TRUE) Or reverse order …	1
2(a)(iii)	IF MainsOn AND (Tray1 OR Tray2) THEN **MP1** **MP2** WashStart ← TRUE **MP3** ENDIF **OR:** **MP2** WashStart ← MainsOn AND (Tray1 OR Tray2) **MP1** **MP3**	3
2(b)(i)	Report window // Watch window	1
2(b)(ii)	103	1

3(a)	**MP1** Investigate the current system /any example
	// investigate any other similar systems
	MP2 **D**iscussion/interviews with managers
	MP3 Discussion/interviews with staff **who operate the system**
	MP4 Carry out a **survey** of **employees/customers**
	// carry out a **focus group of employees/customers**
	MP5 Observe the current system in use
	MP6 Produce the requirements specification
	MP7 Decide on the (**program development) life cycle** to be used
	
	Max 3
		3

Question	Answer	Marks
3(b)	Before any programming is started, the **DESIGN** stage for the proposed new system must be completed. For this large project, the tasks will be broken down into several **MODULES // SUB-TASKS** each will be coded by a different programming team. Once each team has completed its work, **INTEGRATION // UNIT** testing must be carried out. Before the new system is delivered to the business, **ALPHA** testing is carried out in-house. It is decided to install the software initially in only one of the call centres where **BETA** testing is carried out. The new software is finally rolled out to all five call centres. In the first week of its use, staff report that the software still requires extra features which were not originally agreed. The software is withdrawn until **ADAPTIVE** maintenance is completed. 1 mark per gap	**6**
4	*Example solution:* Initialise Count to zero REPEAT INPUT integer number IF integer number > 0 then increment the **count** UNTIL integer number equals 0 OUTPUT Count *Mark as follows:* **MP1** Initialise Count to zero **and** later evidence of use of counter **MP2** Conditional loop **and** terminated when 0 is input **MP3** Test for number > 0 **and** increment count **inside the loop** **MP4** Output the count (either 'sum' or 'count') **Max 4**	**4**

Question	Answer	Marks

Question	Answer	Marks

5(a)	*Example solution:*
	
	`FUNCTION BulbCost(Quantity : INTEGER) RETURNS REAL`
	`  DECLARE Cost : REAL`
	`  DECLARE Twenty : INTEGER`
	`  DECLARE Other : INTEGER`
	
	`  Twenty ← Quantity DIV 20`
	`  Other ← Quantity MOD 20`
	`  Cost ← (Twenty * 30) + (Other * 1.75)`
	`  RETURN Cost`
	`ENDFUNCTION`
	
	Mark as follows:
	MP1 Function header **and** single parameter **and** returns REAL **and** end
	MP2 Declaration of required local variable(s)
	MP3 Correct calculation of 'multiples of 20' and 'other' amounts using MOD and DIV // alternative method
	MP4 Cost is returned
		4

5(b)	`TotalCost ← BulbCost(NumberPurchased)`	**1**

Question	Answer	Marks
6(a)	**MP1** `LEFT(DataPacket, 3)` // `MID(DataPacket, 1, 3)` **MP2** `MID(DataPacket, 5, 1)`	**2**

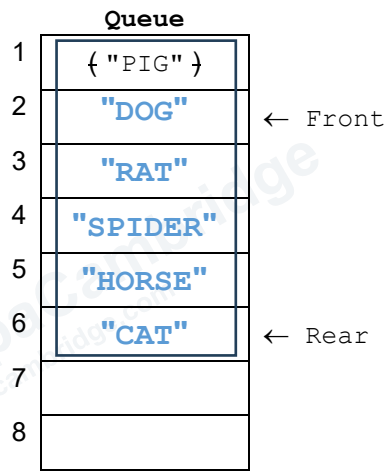
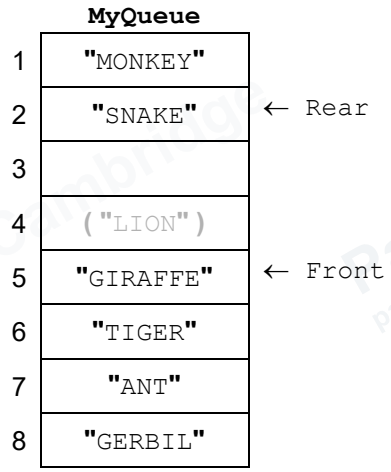
Question	Answer	Marks

6(b)	*Example solution:*
	
	`FUNCTION RejectDescription(ThisReject : STRING)`
	`  RETURNS STRING`
	`  DECLARE Description : STRING`
	`  Description ← MID(ThisReject, 7,`
	Part 1
	`  LENGTH(ThisReject) - 6)`
	Part 2
	OR:
	
	`Description ← RIGHT(ThisReject, LENGTH(ThisReject) - 6)`
	Part 1 *Part 2*
	
	`  RETURN Description`
	`ENDFUNCTION`
	
	Mark as follows:
	
	MP1 Function header **and** end function **(and** string parameter)
	MP2 Calculation for description expression (*part 1*) using `MID/RIGHT`
	MP3 Calculation for description expression (*part 2*) using the `LENGTH` function
	MP4 Return of a calculated value **and** `RETURN STRING` in the header
		4

Question	Answer	Marks

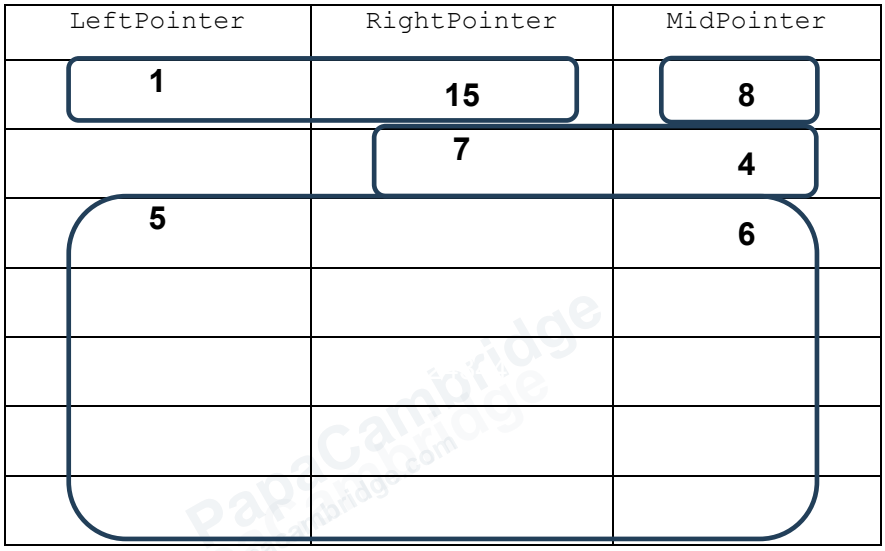
6(c)	*Example solution:*
	
	`DECLARE ThisPacket : STRING`
	`DECLARE ThisLength : STRING`
	`DECLARE ThisMachine : CHAR`
	`DECLARE ThisDescription : STRING`
	
	`OPENFILE "Rejects.txt" FOR READ`
	`WHILE NOT EOF("Rejects.txt")`
	`  READFILE "Rejects.txt", ThisPacket`
	`  ThisLength ← LEFT(ThisPacket, 3)`
	`  ThisMachine ← MID(ThisPacket, 5, 1)`
	`  ThisDescription ← RejectDescription(ThisPacket)`
	`  IF STR_TO_NUM(ThisLength) > 560 AND`
	`  (ThisMachine = 'B' OR ThisMachine = 'C') THEN`
	`  OUTPUT ThisDescription`
	`  ENDIF`
	`ENDWHILE`
	`CLOSEFILE "Rejects.txt"`
	
	Mark as follows:
	MP1 `OPENFILE` in `READ` mode **and** `CLOSEFILE`
	MP2 Conditional loop - terminates when `EOF("Rejects.txt")`
	MP3 `READFILE` – with correct syntax
	MP4 'Attempted' isolation of length **and** machine data
	MP5 `IF STR_TO_NUM(ThisLength) > 560 AND (ThisMachine = 'B' OR ThisMachine = 'C')`
	MP6 `OUTPUT` of return value from `RejectDescription()` **inside the loop**
		6

Question	Answer	Marks
7(a)(i)	Printer queue on a computer network // modelling/simulation of a real-world problem using a queue //accept anything reasonable	**1**
7(a)(ii)	The queue operates on the principle the first item to join the queue will be the next item to leave **MP1** First in – first out 'feel' **MP2** Must refer to their application stated in part **(i)**	**2**
7(b)(i)	**MP1** `DECLARE MyQueue : ARRAY [1:8] OF STRING` **MP2** `DECLARE Front : INTEGER` `DECLARE Rear : INTEGER`	**2**

Question	Answer	Marks
7(b)(ii)	 *Mark as follows:* **MP1** Correct sequence of five values in locations 2–6 **MP2** `Front` points to DOG in location 2 **and** `Rear` points to CAT in location 6	**2**
7(b)(iii)	 Mark as follows: **MP1** MONKEY and SNAKE in locations 1, 2 with correct `Rear` pointer **MP2** GIRAFFE in 5 with `Front` pointer **and** TIGER, ANT, GERBIL in locations 6, 7 and 8	**2**

Question	Answer	Marks

7(c)	Example solution:
	
	`PROCEDURE InitialiseQueue()`
	`  DECLARE i : INTEGER`
	`  Front ← 0 /….7/8`
	`  Rear ← 0 / … 7/8`
	`  FOR i ← 1 TO 8`
	`  MyQueue[i] ← ""`
	`  NEXT i`
	`ENDPROCEDURE`
	
	Mark as follows:
	
	MP1 `Front` and `Rear` are each assigned a value in range
	MP2 Correct `FOR-NEXT` loop // other loop structure
	MP3 `MyQueue[i]` assigned an empty string **in a loop**
		3

8(a)		**4**
8(b)	Found 6	**1**
8(c)	6	**1**

Question	Answer	Marks
9(a)	`DECLARE Version1 : ARRAY [0:2, 1:15] OF CHAR` **MP1** **MP2**	**2**

Question	Answer	Marks

9(b)	*Example solution:*
	
	`DECLARE RoomNumber, FloorNumber, Count : INTEGER`
	
	`Count ← 0`
	`OUTPUT "Floor number: "`
	`INPUT FloorNumber`
	
	`OUTPUT "Floor number: " & NUM_TO_STR(FloorNumber)`
	`FOR RoomNumber ← 1 TO 15`
	`  IF Version1[FloorNumber, RoomNumber] = 'A' THEN`
	`  OUTPUT RoomNumber`
	`  Count ← Count + 1`
	`  ENDIF`
	`NEXT RoomNumber`
	
	`OUTPUT NUM_TO_STR(Count) & " room(s) available"`
	
	Mark as follows:
	
	MP1 Declaration of **all** integer variables
	MP2 Input of the floor number (with prompt)
	MP3 Loop for 15 iterations
	MP4 Test array element to check if room is 'available'?
	MP5 Initialisation **and** increment of count **and** Output room number
	MP6 Output of the final count line including the message **outside the loop**
		6

9(c)(i)	***Example solution:***
	
	`TYPE RoomData`
	`  (DECLARE Available : CHAR)`
	`  DECLARE RoomType : CHAR`
	`  DECLARE View : BOOLEAN`
	`  DECLARE Grade : STRING`
	`ENDTYPE`
	
	Mark as follows:
	
	MP1 and MP2
	Two variables correct 1 mark
	Three variables correct 2 marks
	MP3 `ENDTYPE`
		3

Question	Answer	Marks

9(c)(ii)	*Example solution:*
	`DECLARE BoolView : BOOLEAN`
	`DECLARE MyRoomType : CHAR`
	`DECLARE InputView, MyRoomGrade : STRING`
	`DECLARE Floor, Room : INTEGER`
	
	`OUTPUT "Room type - S/D/F ?"`
	`INPUT MyRoomType`
	
	`OUTPUT "View - True/False ?"`
	`INPUT InputView`
	`IF TO_UPPER(InputView) = "TRUE" THEN`
	`  BoolView ← TRUE`
	`ELSE`
	`  BoolView ← FALSE`
	`ENDIF`
	
	`OUTPUT "Room grade - ST/DL ?"`
	`INPUT MyRoomGrade`
	
	`OUTPUT "Matching floor number and room number:"`
	`FOR Floor ← 0 TO 2`
	`  FOR Room ← 1 TO 15`
	`  IF Version2[Floor, Room].RoomType = MyRoomType`
	`  AND Version2[Floor, Room].View = BoolView`
	`  AND Version2[Floor, Room].RoomGrade = MyRoomGrade`
	`  AND Version2[Floor, Room].Available = 'A'THEN`
	`  OUTPUT Floor,"  ",Room`
	`  ENDIF`
	`  NEXT Room`
	`NEXT Floor`
	
	Marks as follows:
	MP1 **Three** correctly formed INPUT statements each with a prompt
	
	MP2 `OUTPUT` of the "Matching floor etc" heading
	MP3 Outer FOR loop / or other (for the floors)
	MP4 Inner FOR loop / or other (for the rooms)
	
	MP5 Correct use of array with dot notation with at least one data item **inside loop**
	MP6 'Attempt' to combine all three tests (room type, view, grade) **inside loop**
	MP7 Output of floor and room **inside loop**
		7