CAMBRIDGE
International Education

Cambridge International AS & A Level

COMPUTER SCIENCE **9618/23**

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 **15** 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	Guidance
1(a)	Post-condition(al) loop // post-condition(al) iteration	1	
1(b)	5	1	

Question	Answer	Marks	
2(a)	`BOOLEAN`	1	

2(b)	`IF MainsOn (=TRUE) THEN`
	`  DisplayOn ← TRUE`
	`ENDIF`
	Accept:
	`DisplayOn ← MainsOn`
	
	Accept '=' in place of the assignment arrow
		1	

2(c)	**MP1** The robot is only ready to move if… the **mains power is on and** ….. **MP2** The strength of the (**suction) value selected is non-zero**	2	Reject – Zero for a 'repeat' of the pseudocode statement.
2(d)(i)	breakpoint	1	
2(d)(ii)	38 (minutes)	1	

Question	Answer	Marks	
3(a)	The initial **ANALYSIS** stage is carried out by studying in detail the current system. A design for the proposed new system is produced. Following the design and coding stages, it is decided to install the completed new software initially in only two of the shops**.** **BETA** testing is carried out. The final stage of the project uses **ACCEPTANCE** testing to test all programs on the company's hardware. After successful testing, the company approves the new system. The company then installs the new software in all shops. In the first week of its use, unexpected bugs are reported. **CORRECTIVE** maintenance has to be carried out.	4	

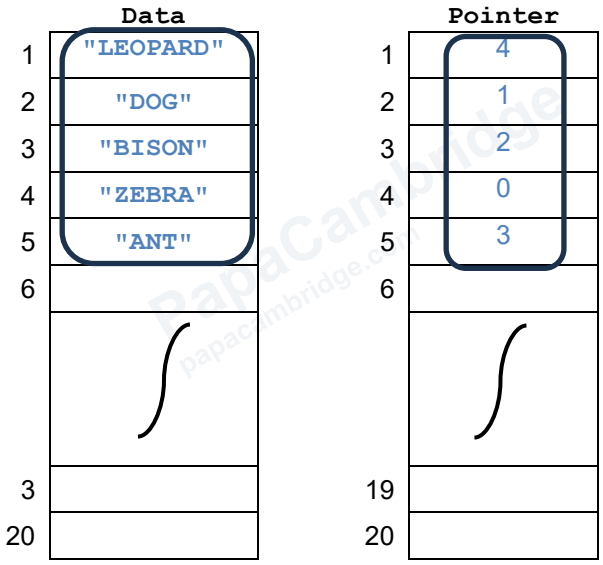
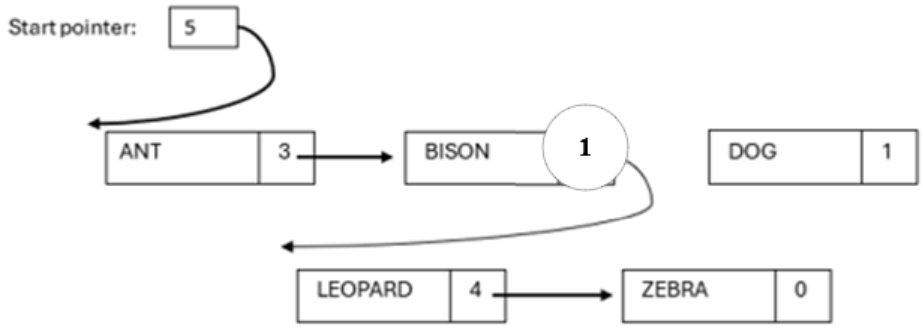
Question	Answer	Marks	Guidance
3(b)	**Team A** **MP1** Rapid Application Development /RAD // Iterative **MP2** Large project can be broken down into smaller steps **MP3** All stakeholders/users are involved (at all stages) **MP4** There is constant discussion between the client (and Team A) **MP5** Prioritises the client needs and end-user experience **MP6** A prototype is available at an early stage **MP7** More readily allows for **requirement changes** to be made **MP8** Minimal planning is required **MP9** Programs/modules can be developed in parallel **Max 2** **Team B** **MP10** Waterfall (method/model) **MP11** Next stage does not start until the current stage is successfully completed **MP12** More predictable for budget **MP13** More predictable for timeline/completion **MP14** All requirements are identified before coding **MP15** Late changes cannot be accommodated **Max 2**	**4**	Reject 'early version'.
3(c)	**MP1** Production of flowcharts // structure charts // STDs // Structured English // E-R diagrams **MP2** Consider data storage (hardware / database / files) **MP3** Design inputs screens // Design screen outputs // reports to be produced **MP4** Select (Programming) language to be used **MP5** Produce pseudocode algorithms / data structures / procedures/functions/modules are produced / Identifier table **MP6** Produce a testing strategy statement // (detailed) test plan **MP7** Consider – is there relevant code available from **program libraries**? **Max 3**	**3**	Reject 'program structure'.

Question	Answer	Marks	Guidance
4	*Example solution:* • Initialise count to zero • REPEAT • Input next number • If number > 120 then increment the count • Until number = -1 • Output the count *Mark as follows:* **MP1** Initialise count **and** later evidence of use of counter (**not** 'sum') **MP2** Conditional loop and terminated when –1 is input **MP3** Test for number > 120 **and** increment the count (**not** 'sum') **inside the loop** **MP4** Output the count (Accept either 'sum' or 'count')	**4**	A 'sum' solution will have a Max 2.

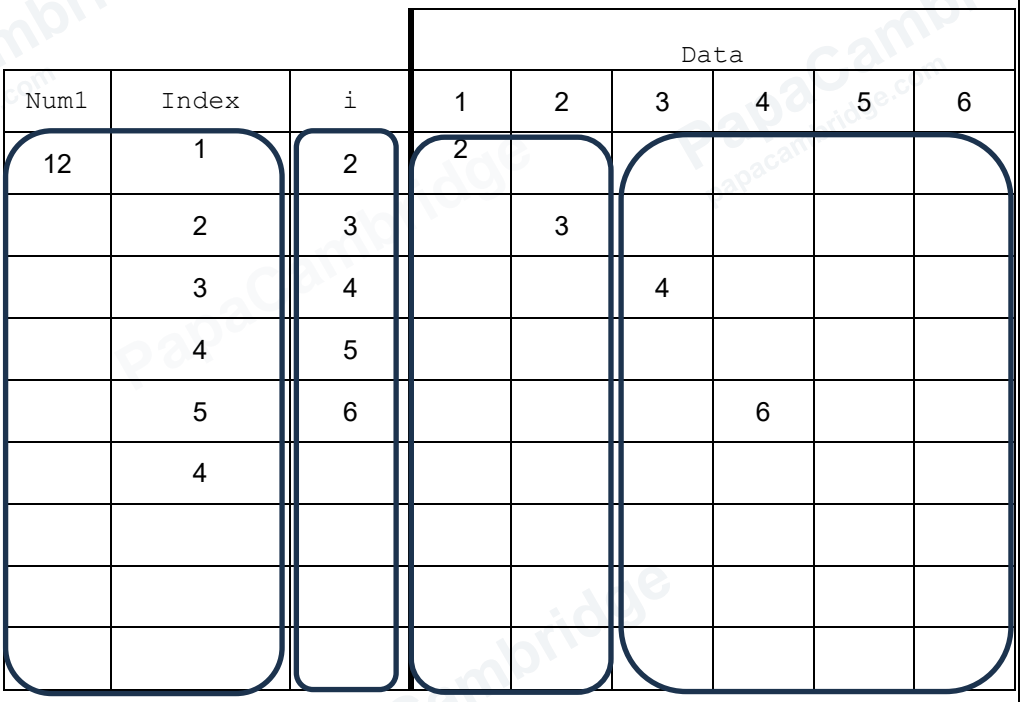
5(a)	`FUNCTION InterviewSummary(ThisApplicant : STRING)`
	`RETURNS STRING`
	`DECLARE Summary : STRING`
	
	`Summary ← MID(ThisApplicant, 8,`
	MP2
	`LENGTH(ThisApplicant)- 7)`
	MP3
	Or:
	`Summary ←RIGHT(ThisApplicant, LENGTH(ThisApplicant) – 7)`
	~~**MP2**~~ ~~**MP3**~~
	`RETURN Summary`
	`ENDFUNCTION`
	
	Mark as follows:
	
	MP1 Function identifier **and** function ending **and** parameter
	
	MP2 Calculation for summary expression (part 1) – using `MID` or `RIGHT`
	MP3 Calculation for summary expression (part 2) – using `LENGTH`
	MP4 Return of the calculated `STRING` value (earlier declaration optional) **and** `RETURNS STRING` in the header –
		4	Do not accept a final **missing close bracket** (for the `MID` or `RIGHT` function).

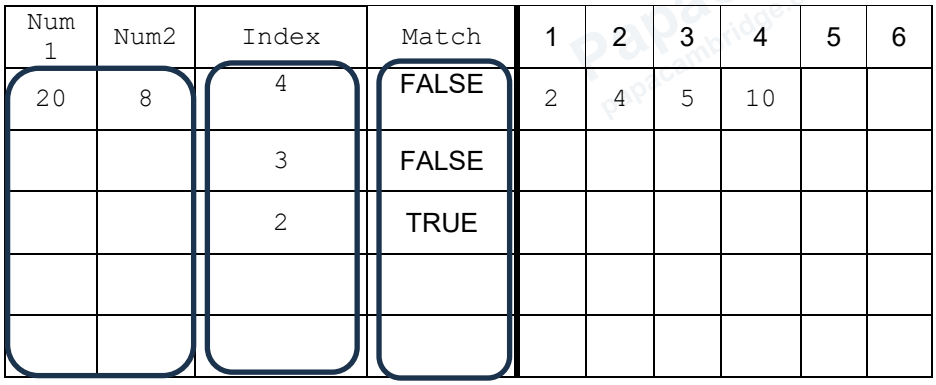
Question	Answer	Marks	Guidance

5(b)	`PROCEDURE Search(Initial : CHAR, ShortSurname : STRING)`
	`  DECLARE ThisApplicant, ThisSurname : STRING`
	`  DECLARE ThisInitial : CHAR`
	`  DECLARE Found : BOOLEAN`
	`  OPENFILE "JobApps.txt" (FOR) READ`
	`  Found ← FALSE`
	`  REPEAT`
	`  READFILE "JobApps.txt", ThisApplicant`
	
	`  ThisInitial ← LEFT(ThisApplicant, 1)`
	`  ThisSurname ← MID(ThisApplicant, 2, 5)`
	`  IF ThisSurname = ShortSurname`
	`  AND ThisInitial = Initial THEN`
	`  Found ← TRUE`
	`  OUTPUT "Name found"`
	`  ENDIF`
	`  UNTIL Found = TRUE OR EOF("JobApps.txt")`
	
	`  IF Found = FALSE THEN`
	`  OUTPUT "Name not Found"`
	`  ENDIF`
	`  CLOSE(FILE) "JobApps.txt"`
	
	`ENDPROCEDURE`
	
	Mark as follows:
	1 mark per gap
		7	

Question	Answer	Marks	Guidance
6(a)	**MP1** 5  **MP2** Correct `Data` array values **MP3** Correct pointers	**3**	
6(b)	The end of **'(linked) list'**	**1**	
6(c)(i)	 *Mark as follows:* Start is 5 **and** other pointer values are correct **and** BISON pointer is changed to 1 **and** arrows are as shown **and** DOG is 'isolated'	**1**	
6(c)(ii)	**MP1** The item is not in the list **MP2** The data item is the only item in the list **MP3** The item is the first in the list	**2**	

Question	Answer	Marks	Guidance
6(d)	**MP1** Store 'BAT' at the next available location /with index 6 / in the `Data` array **MP2** Visit the start pointer node **MP3** Compare this data value with 'BAT' //compare 'BAT' with 'ANT' **MP4** When 'BAT' is less than the current value - 'position is found' **MP5** IF not, move to next node // move to 'BISON' **MP6** ….. and store the previous position **MP7** Repeat until 'BAT' less that current value **MP8** Link pointer of the previous node // 'ANT' points/connects to 'BAT' **MP9** Link pointer of 'BAT' points to the current node / 3 // 'BAT' points/connects to 'BISON' **Max 6**	**6**	

Question	Answer	Marks	Guidance
7(a)(i)	 Mark by column(s)	**4**	
7(a)(ii)	Factors / the divisors of `Num1/12` // all the numbers which divide exactly into `Num1`	**1**	

Question	Answer	Marks	Guidance
7(b)(i)	 **Max 2** for any additional incorrect entries	**3**	
7(b)(ii)	Highest common factor // Greatest common divisor of `Num1` and `Num2` // by good explanation i.e. 4 is the **highest** number that divides exactly into **both** 8 and 20	**1**	

Question	Answer	Marks	Guidance
8(a)(i)	`DECLARE Bins : ARRAY [0:2, 1:120] OF BOOLEAN` **MP1** **MP2**	**2**	Do not accept variations of syntax.

Question	Answer	Marks	Guidance

8(a)(ii)	*Example solution:*
	
	`PROCEDURE InactiveReport()`
	`  DECLARE Level, BinNo : INTEGER`
	`  DECLARE BinCount : INTEGER`
	`  BinCount ← 0`
	`  FOR Level ← 0 TO 2`
	`  IF Level = 0 THEN`
	`  OUTPUT "Level: ground floor"`
	`  ELSE`
	`  OUTPUT "Level: " & NUM_TO_STR(Level)`
	`  ENDIF`
	`  FOR BinNo ← 1 TO 5`
	`  IF Bins[Level, BinNo] = FALSE THEN`
	`  OUTPUT "Bin " & NUM_TO_STR(BinNo)`
	`  BinCount ← BinCount + 1`
	`  ENDIF`
	`  NEXT BinNo`
	`  NEXT Level`
	`  OUTPUT NUM_TO_STR(BinCount) & " bins are not in use"`
	`ENDPROCEDURE`
	
	Mark as follows:
	MP1 Bin count declared **and** initialised before the loop **and** correctly incremented
	
	MP2 (Outer) level loop – iterates 3 times
	
	MP3 **(**Inner) bin loop – iterates 5 times
	
	MP4 Correct 2D array syntax **and** test for not in use with a Boolean comparison
	
	MP5 **C**orrect **header** OUTPUT statement(s) - includes the ‘Ground floor‘ text **and** ‘Bin’ text **and** bin number
	MP6 Correct final OUTPUT statement with ‘total not in use’ text **and** the count
	
	Max 5
		5	

Question	Answer	Marks	Guidance

8(a)(iii)	`Bins[0, 3] ← FALSE`
	
	`Bins[1, 2] ← TRUE`
	
	`Bins[2, 1] ← FALSE`
	`Bins[2, 2] ← FALSE`
	`Bins[2, 3] ← TRUE`
	`Bins[2, 4] ← TRUE`
	`Bins[2, 5] ← FALSE`
	
	Mark as follows:
	
	MP1 `Bins(0, 3) ← FALSE`
	MP2 `Bins(1, 2) ← TRUE`
	
	MP3 Level 2 assignment all complete
		3	

8(b)(i)	*Example solution:*
	
	`TYPE BinData`
	
	`  DECLARE Size : STRING`
	`  DECLARE Active : BOOLEAN`
	`  DECLARE StockCode : STRING`
	`  DECLARE StockLevel : INTEGER`
	
	`ENDTYPE`
	
	Mark as follows:
	
	Two variables correct – 1 Mark
	4 variables correct – 2 marks
		2	

Question	Answer	Marks	Guidance

8(b)(ii)	*Example solution:*
	
	`FUNCTION FindBin(MySize : STRING) RETURNS STRING`
	`  DECLARE Level, BinNo : INTEGER`
	`  DECLARE Text : STRING`
	`  Text ← "No bins available"`
	
	`  Level ← 0`
	`  REPEAT`
	`  BinNo ← 1`
	`  REPEAT`
	`  IF Bins[Level, BinNo].Size = MySize`
	`  AND Bins[Level, BinNo].Active = FALSE THEN`
	`  Text ← "Level: " & NUM_TO_STR(Level)`
	`  & " Bin: " & NUM_TO_STR(BinNo)`
	`  OUTPUT Text`
	`  RETURN Text`
	`  ELSE`
	`  BinNo ← BinNo + 1`
	`  ENDIF`
	`  UNTIL BinNo = 121`
	`  Level ← Level + 1`
	`  UNTIL Level = 3`
	
	`  OUTPUT Text`
	`  RETURN Text`
	`ENDFUNCTION`
	
	Mark as follows:
	
	MP1 Function header **and** string parameter **and** return string **and** ending
	
	MP2 Use of any (outer level) loop for 3 iterations
	MP3 Use of any (inner) bin loop for 5 / 120 iterations
	
	MP4 Correct 2D array syntax with either data field – `.Size` or `.Active`
	MP5 Correct IF statement condition including `AND`
	
	MP6 Formation of the text string with correct syntax –
	
	MP7 Correct return value (and output) – **in both cases** and in the correct place
		7	Do not accept comma syntax.