

Cambridge International AS & A Level

COMPUTER SCIENCE**9618/22**

Paper 2 Fundamental Problem-solving and Programming Skills

October/November 2025

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 2025 series for most Cambridge IGCSE, Cambridge International A and AS Level components, and some Cambridge O Level components.

This document consists of **15** 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 descriptions 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.

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

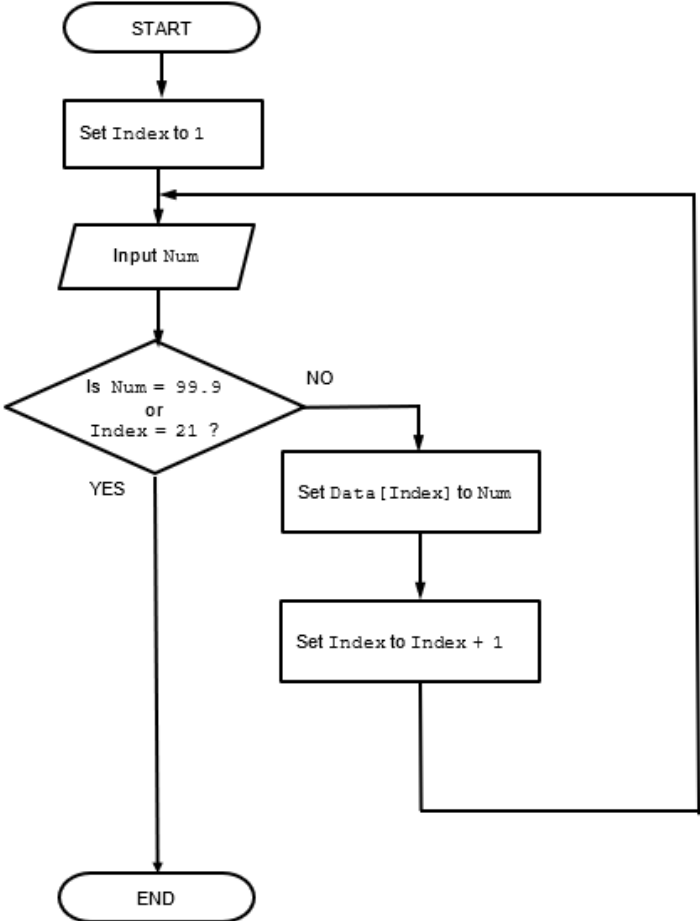
Annotation	Meaning
	Benefit of the doubt
	To indicate where a key word/phrase/code is missing
	Incorrect
	Follow through
	Indicate a point in an answer
Highlighted text	To draw attention to a particular aspect or to indicate where parts of an answer have been combined
	Ignore
	Not answered question
	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 on a page has been seen including blank answer spaces and blank pages.
	Correct
	Too vague

Mark scheme abbreviations

/	separates alternative words / phrases within a marking point
//	separates alternative answers within a marking point
underline	actual word(s) 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

Question	Answer				Marks
1(a)	Pseudocode example	Selection	Iteration	Subroutine	4
	IF Status = FALSE THEN FOR Count ← 1 TO 20 CALL Reset (Count) NEXT Count ENDIF	✓	✓	✓	
	OTHERWISE : Status ← TRUE	✓			
	WHILE AllDone() = TRUE		✓	✓	
	30 : NextChar ← 'X'	✓			
One mark per row					
1(b)	Variable	Example data value	Data type		3
	Result	5.42	REAL		
	MonthLetter	"JFMAMJJASOND"	STRING		
	Birthday	15/11/2009	DATE		
One mark per row					
1(c)	Expression	Evaluates to			4
	INT(Result) + 1 > 6	FALSE			
	NUM_TO_STR(LENGTH (MonthLetter))	"12"			
	NUM_TO_STR(Result + "3.2")	ERROR			
	MID(MonthLetter, MONTH(Birthday) - 2, 1)	"S"			
One mark per row					

Question	Answer	Marks
2	<pre> graph TD START([START]) --> SetIndex[Set Index to 1] SetIndex --> InputNum[/Input Num/] InputNum --> IsNum{Is Num = 99.9?} IsNum -- No --> SetData[Set Data[Index] to Num] SetData --> SetIndexInc[Set Index to Index + 1] SetIndexInc --> IsIndex{Is Index = 21?} IsIndex -- No --> InputNum IsIndex -- Yes --> END([END]) IsNum -- Yes --> END </pre> Mark as follows: MP1 Initialisation of the array index MP2 Input number in a loop MP3 Test for termination value (99.9) and correct ending MP4 Test for array full / maximum 20 iterations and correct ending MP5 Assign number to <u>Data</u> array element and increment index	5

Question	Answer	Marks
2	Alternative solution:  <pre> graph TD Start([START]) --> SetIndex[Set Index to 1] SetIndex --> InputNum[/Input Num/] InputNum --> Decision{Is Num = 99.9 or Index = 21 ?} Decision -- YES --> End([END]) Decision -- NO --> SetData[Set Data[Index] to Num] SetData --> SetIndexInc[Set Index to Index + 1] SetIndexInc --> InputNum </pre>	

Question	Answer			Marks	
3(a)	Mark as follows:			6	
	MP1	1			open the file OldFile.txt in read (mode)
	MP2	2			open the file NewFile.txt in write (mode)
		3	}		read a line from OldFile.txt and store in LineX
	MP3	4	}		read a line from OldFile.txt and store in LineY
		5	}		read a line from OldFile.txt and store in LineZ
	MP4	6			set NewString to LineX & '\ ' & LineY & '\ ' & LineZ
	MP5	7			write NewString to NewFile.txt
	MP6	8			repeat from step 3 until end of OldFile.txt
		9			close both files
3(b)	It does not appear in (any of) the data items // it does not appear in the ID and Name			1	
3(c)	MP1 MP2 MP3 MP4 Max 3	Add a (single) new separator at the end of the third (original) item Calculate the length of both new data items Store / append the length (of the new each items) as a fixed length / separated digit string // by example		3	

Question	Answer	Marks
4(a)	Example solution: <pre> DECLARE CheckTotal : ARRAY [0:35] OF BOOLEAN DECLARE Index : INTEGER FOR Index ← 0 TO 35 CheckTotal[Index] ← FALSE NEXT Index </pre> Mark as follows: MP1 Declaration of <code>CheckTotal</code> array MP2 Declaration of a loop counter MP3 Loop – with correct syntax and number of iterations MP4 Assignment of array element to <code>FALSE</code>	4
4(b)	<pre> DECLARE EasyQ, HardQ : INTEGER FOR EasyQ ← 0 TO 15 STEP 3 MP1 MP2 FOR HardQ ← 0 TO 20 STEP 5 MP3 MP4 CheckTotal[HardQ + EasyQ] ← TRUE NEXT HardQ MP5 NEXT EasyQ </pre> Mark as follows: One mark per gap (boldened)	5
4(c)	MP1 Check that the parameter value is in the range 0 to 35 MP2 ... and if not, return <code>FALSE</code> MP3 Use the parameter value as the index to array <code>CheckTotal</code> MP4 Return the value from given index position	4

Question	Answer		Marks
5(a)	Activity	Name of life cycle stage	4
	a structure chart is produced	Design	
	a program is modified to allow it to run on new hardware	Maintenance	
	the programmer identifies the customer's requirements	Analysis	
	an Integrated Development Environment (IDE) provides context-sensitive help such as 'auto-complete'	Coding	
	One mark per row		
5(b)	Acceptance testing		1

Question	Answer	Marks
6	Example solution: <pre> FUNCTION Compare(String1, String2 : STRING, __ CaseMatters : BOOLEAN, Position : CHAR) RETURNS BOOLEAN DECLARE S1Length : INTEGER S1Length ← LENGTH(String1) IF LENGTH(String2) < S1Length THEN RETURN FALSE // String2 is shorter than String1 ENDIF IF CaseMatters = FALSE THEN String1 ← TO_UPPER(String1) String2 ← TO_UPPER(String2) ENDIF CASE Position 'e' : String2 ← RIGHT(String2, S1Length) 's' : String2 ← LEFT(String2, S1Length) ENDCASE IF String1 = String2 THEN RETURN TRUE ELSE RETURN FALSE ENDIF ENDFUNCTION </pre> Mark as follows: MP1 Function heading and parameters and ending and return type MP2 Calculate length of String1 MP3 if String2 is shorter than String1, return FALSE MP4 Compare (modified) strings MP5 Test CaseMatters - if FALSE and convert two string(s) to same case MP6 Test for the value of 's'/'e' and form modified string(s) in one or both cases MP7 Return appropriate BOOLEAN value in all cases	7

Question	Answer	Marks
7(a)	Example explanation: The arrow means that <code>Convert</code> repeatedly calls <code>Analyse</code> then <code>Update</code> and then <code>Sort</code> (in that order)¹⁰ MP1 Reference to 'repetition'... MP2 Naming all <u>four</u> modules correctly and the sequence is stated or implied	2
7(b)	MP1 <u>PROCEDURE Analyse (Count : INTEGER)</u> <u>FUNCTION Update (Key : STRING) RETURNS INTEGER</u> MP2 MP3 MP4 <u>PROCEDURE Sort (BYREF P2 : BOOLEAN)</u> Mark as follows: One mark per underlined part	4

Question	Answer	Marks
8(a)	Example solution: <pre> PROCEDURE CountLoans(ThisStudentID : STRING) DECLARE Index, LCount, RCount : INTEGER LCount ← 0 RCount ← 0 FOR Index ← 1 TO 7000 IF Loan[Index].StudentID = ThisStudentID THEN IF Loan[Index].OnLoan = FALSE THEN RCount ← RCount + 1 ELSE LCount ← LCount + 1 ENDIF ENDIF NEXT Index OUTPUT "Student has ", LCount, " books on loan" OUTPUT "and has returned ", RCount, " books" ENDPROCEDURE </pre> Mark as follows: MP1 Declaration of Index and two count variables MP2 Loop for 7000 iterations MP3 Index references an indexed data item with correct .dot notation MP4 Attempt to compare the StudentID field = parameter in a loop MP5 Test of the OnLoan field and correct logic for both outcomes MP6 Both count variables initialised and increment appropriate count in a loop MP7 Output includes both counts with a <u>suitable</u> message	7

Question	Answer	Marks
8(b)	Example solution: <pre> FUNCTION NewLoan(ThisStudentID, ThisBookID : STRING) RETURNS BOOLEAN DECLARE Index : INTEGER DECLARE IsStored : BOOLEAN CONSTANT Blank = "" Index ← 1 IsStored ← FALSE WHILE Index <= 7000 AND NOT IsStored IF Loan[Index].StudentID = Blank THEN Loan[Index].StudentID ← ThisStudentID Loan[Index].BookID ← ThisBookID Loan[Index].OnLoan ← TRUE IsStored ← TRUE ENDIF Index ← Index + 1 ENDWHILE RETURN IsStored ENDFUNCTION </pre> Mark as follows: MP1 Loop through 7000 elements in <code>Loan</code> array – correct loop structure MP2 or until the first unused element is found // Correct <code>RETURN</code> inside a <code>FOR</code> loop MP3 Correct use of the indexed dot notation MP4 Test if record is unused (<code>.StudentID = ""</code>) in a loop MP5 Two item assignment statements correct MP6 <code>Loan[Index].OnLoan</code> assigned <code>TRUE</code> MP7 Correct logic to return <code>BOOLEAN</code> in both cases (position found / no position available) Alternative solution: loop only (no ‘found’ mechanism) <pre> FOR Index ← 1 TO 7000 IF Loan[Index].StudentID = Blank THEN Loan[Index].StudentID ← ThisStudentID Loan[Index].BookID ← ThisBookID Loan[Index].OnLoan ← TRUE RETURN TRUE ENDIF NEXT RETURN FALSE </pre>	7

Question	Answer	Marks
8(c)	MP1 Save to/Open a (text) file MP2 Single line – form a string with separator characters and write to the file // Multiple lines - write each data item to a separate line in the file	2
8(d)	Additional date(s) solution MP1 Store the return date of each loan // the start date <u>and</u> the loan period MP2 Algorithm will do a calculation involving the MP1 data item(s) OR The student email address solution MP3 Store the student <u>email address</u> MP4 In order to send the email // the student is alerted to return the book Max 2	2