Cambridge Assessment International Education

Cambridge IGCSE™ (9–1)

COMPUTER SCIENCE **0984/02**

Paper 2 Algorithms, Programming and Logic **For examination from 2023**

MARK SCHEME

Maximum Mark: 75

Specimen

This document has **16** pages. Any blank pages are indicated.

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:

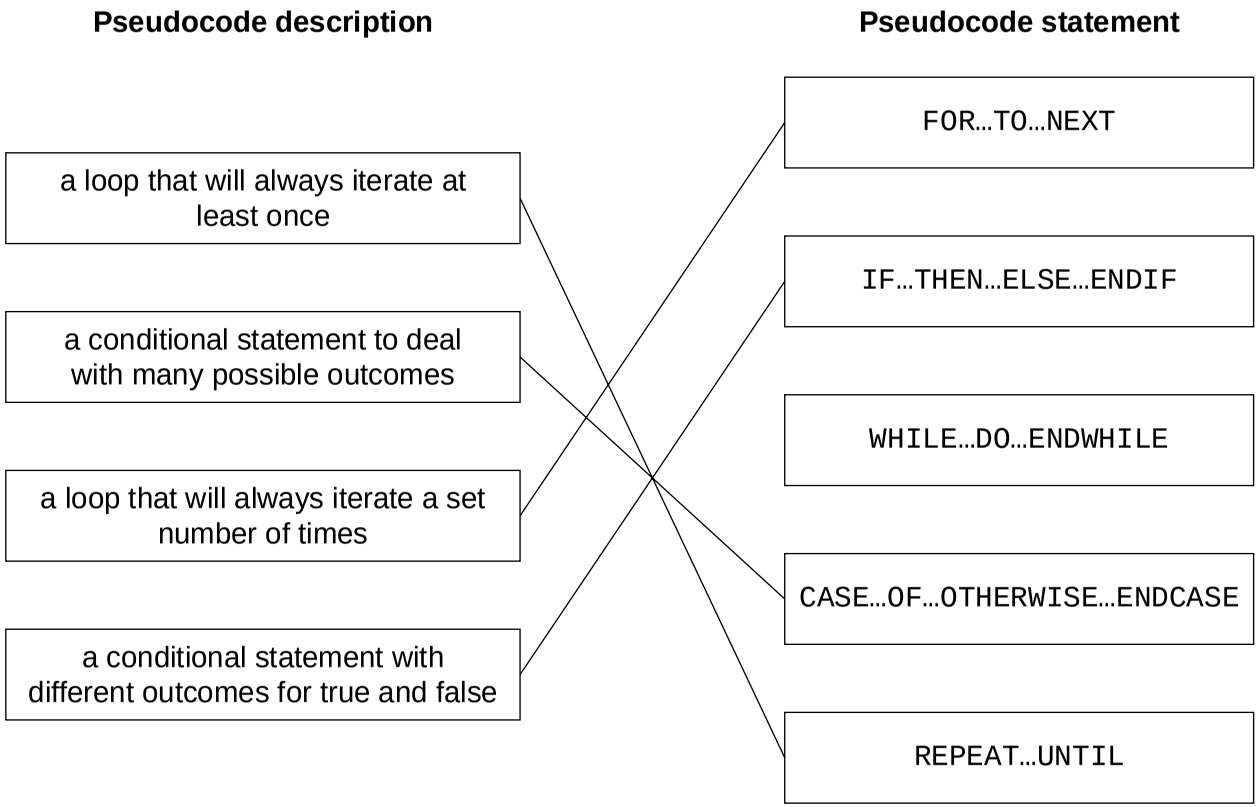
Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

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)	**One** mark for each correct line.	**4**

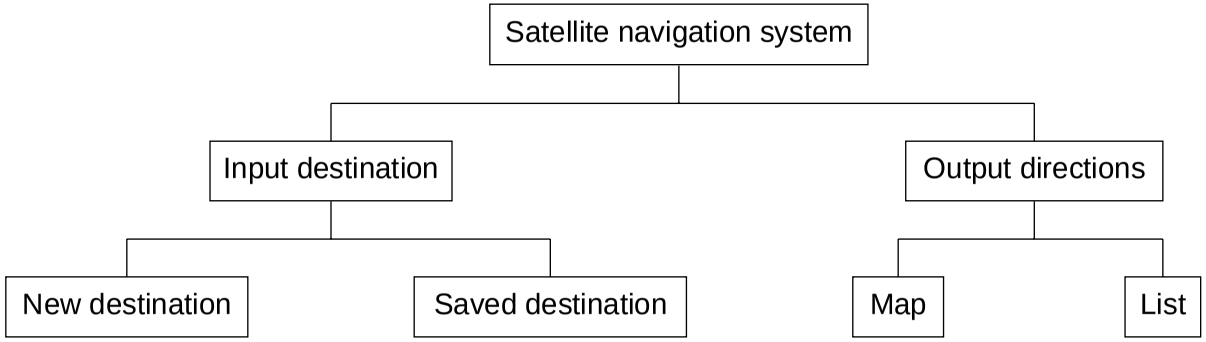


Question	Answer	Marks

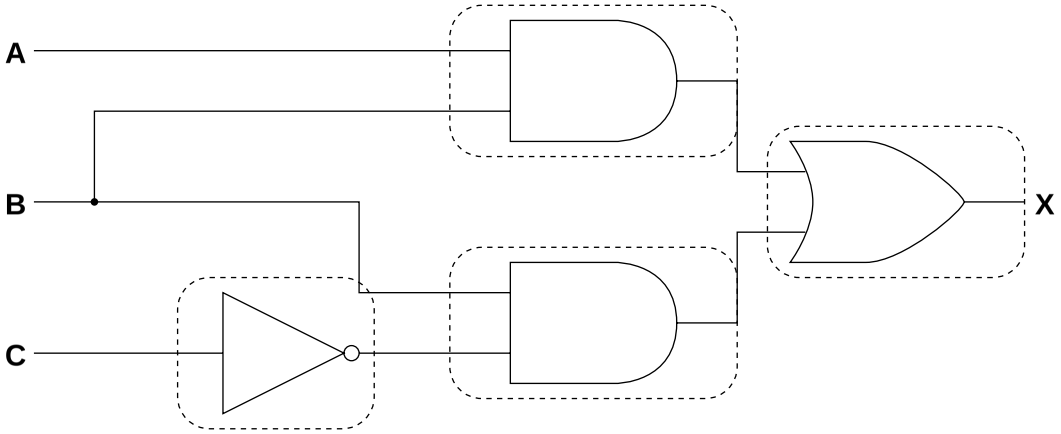
1(b)	• appropriate loop controls
	• read from array
	• output from array (the last two points can be in one statement, see example)
	
	Note: reading and the output **MUST** be within the same loop.
	
	For example:
	`Count ← 0`
	`WHILE Count < 50 DO`
	`  OUTPUT Name[Count]`
	`  Count ← Count + 1`
	`ENDWHILE`
		3

Question	Answer	Marks
2	**Validation check** **One** mark for description: • To test if the data entered is possible / reasonable / sensible. • A range check tests that data entered fits within specified values. **One** mark for example: • Allow any correct validation check as an example (range, length, type, presence, format, etc.). **Verification check** **One** mark for description: • To test if the data input is the same as the data that was intended to be input. • A double entry check expects each item of data to be entered twice and compares both entries to check they are the same. **One** mark for example: Allow any correct verification check as an example (visual, double entry, etc.).	**4**

Question	Answer	Marks
3	**B**	**1**

Question	Answer	Marks
4	**One** mark for a hierarchical structure. **One** mark for suitable names for the sub-systems. **One** mark for identifiable inputs. **One** mark for identifiable outputs. For example:  	**4**

Question	Answer	Marks
5(a)	**One** mark for each error identified and correction: • `Numbers` should be `Number` • `IF Number > 100` should be `IF Number >= 100` • `INPUT Number` is missing from inside the loop, insert `INPUT Number` after the `ENDIF` statement. • The final `OUTPUT Number` is not needed, remove it.	**4**
5(b)	**One** mark for both ends of the range and correct inequality symbols. **One** mark for the **AND** // nested IFs. The test should be `IF Number >= 100 AND Number <= 200`	**2**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>6(a)</td><td>One mark for each correct gate, with the correct input(s) as shown.
 
</td><td>4</td></tr>
<tr><td>6(b)</td><td>
<table>
<tr><th>A</th><th>B</th><th>C</th><th>X</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td></tr>
</table>
Four marks for eight correct outputs.
Three marks for six or seven correct outputs.
Two marks for four or five correct outputs.
One mark for two or three correct outputs.</td><td>4</td></tr>
</table>

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>7(a)</td><td>
<table>
<tr><th>Flag</th><th>Count</th><th>Name[1]</th><th>Name[2]</th><th>Name[3]</th><th>Name[4]</th><th>Temp</th></tr>
<tr><td></td><td></td><td>Jamal</td><td>Amir</td><td>Eve</td><td>Tara</td><td></td></tr>
<tr><td>0</td><td>1</td><td>Amir</td><td>Jamal</td><td>Eve</td><td>Tara</td><td>Jamal</td></tr>
<tr><td>1</td><td>2</td><td>Amir</td><td>Eve</td><td>Jamal</td><td>Tara</td><td>Jamal</td></tr>
<tr><td>1</td><td>3</td><td>Amir</td><td>Eve</td><td>Jamal</td><td>Tara</td><td>Jamal</td></tr>
<tr><td>1</td><td>4</td><td>Amir</td><td>Eve</td><td>Jamal</td><td>Tara</td><td>Jamal</td></tr>
<tr><td>0</td><td>1</td><td>Amir</td><td>Eve</td><td>Jamal</td><td>Tara</td><td>Jamal</td></tr>
<tr><td>0</td><td>2</td><td>Amir</td><td>Eve</td><td>Jamal</td><td>Tara</td><td>Jamal</td></tr>
<tr><td>0</td><td>3</td><td>Amir</td><td>Eve</td><td>Jamal</td><td>Tara</td><td>Jamal</td></tr>
<tr><td>0</td><td>4</td><td>Amir</td><td>Eve</td><td>Jamal</td><td>Tara</td><td>Jamal</td></tr>
</table>

One mark for Flag column.
One mark for Count column.
One mark for Temp column.
Two marks for all correct Name columns or one mark for two or three correct Name columns.

Note: Repeated values do not need to be written unless a value is rewritten.
</td><td>5</td></tr>
<tr><td>7(b)</td><td>• bubble sort / sorting the names
• ascending order / A to Z / lowest to highest / alphabetical order</td><td>2</td></tr>
</table>

Question	Answer	Marks
8	**One** mark for each. *10.00* boundary / abnormal data // the price should be rejected // value is out of range *9.99* boundary / extreme / normal data // the price should be accepted // value is within normal range *ten* abnormal data // input should be rejected // value is wrong type	**3**

Question	Answer	Marks
9	Any **three** from: • data is not lost when the computer is switched off // data is stored permanently • data can be used by more than one program or reused when a program is run again • data can be backed up or archived • data can be transported from one place / system to another.	**3**

Question	Answer	Marks
10	**C**	**1**

Question	Answer	Marks
11(a)	20	**1**
11(b)(i)	`CatNo`	**1**
11(b)(ii)	it is a unique identifier	**1**
11(c)	**Two** marks for four correct answers. **One** mark for two or three correct answers. **Field** / **Data type** `CatNo` – Text `Title` – Text `Genre1` – Text `Streaming` – Boolean / Text	**2**
11(d)	`FROM` `"Comedy"`	**2**

Question	Answer	Marks

12(a)	**One** mark for each correct line.
	
	`DECLARE X : STRING`
	`DECLARE Y : INTEGER`
	`DECLARE Z : INTEGER`
		3

12(b)	**One** mark for storing string in `X`.
	One mark for calling the function length.
	One mark for using the correct parameter `X`.
	One mark for using the substring function.
	One mark for correct parameters.
	One mark for outputting length and substring return values.
	For example:
	`X ← "Programming is fun"`
	`OUTPUT Length(X)`
	`Y ← 16`
	`Z ← 3`
	`OUTPUT SubString(X,Y,Z)`
		6

Question	Answer	Marks
13	Read the whole answer, award a mark from both of the following tables and add up the total. Marks are available for: • AO2 (maximum 9 marks) • AO3 (maximum 6 marks). The techniques and the data structures required are listed below. The requirements may be met using a suitable built-in function from the programming language used (e.g. Python, VB.NET or Java). **Techniques required:** **R1** Calculate total mark for each student (iteration and totalling). **R2** Calculate average mark for each student rounded to the nearest whole number. **R3** Selection of grade for each student (selection). **R4** Output for each student name, total mark, average mark, grade awarded (output with appropriate messages). **R5** Calculate, store and output the number of distinctions, merits, passes and fails for the whole class (iteration, counting and output with appropriate messages). **Data structures required:** The names underlined must be used as provided in the scenario. Arrays or lists `StudentName`, `StudentMark`, (`TotalMark` and `AverageMark` may be seen but no requirement to store) Variables `ClassSize`, `SubjectNo`, `SubjectCounter`, `StudentCounter` `DistinctionNo`, `MeritNo`, `PassNo`, `FailNo` could be an array or list Constants `Distinction`, `Merit`, `Pass` could be variables	**15**

Question	Answer	Marks
13	***Example 15 mark answer in pseudocode.***	

```
// meaningful identifier names and appropriate data structures (variables, constants and the
// given arrays) to store all the data required
DECLARE TotalMark : ARRAY[1:50] OF INTEGER
DECLARE AverageMark : ARRAY[1:50] OF INTEGER
DECLARE SubjectCounter : INTEGER
DECLARE StudentCounter : INTEGER
DECLARE DistinctionNo : INTEGER
DECLARE MeritNo : INTEGER
DECLARE PassNo : INTEGER
DECLARE FailNo : INTEGER

CONSTANT Distinction = 70
CONSTANT Merit = 55
CONSTANT Pass = 40

// initialisation processes for this scenario, initialising the running totals used for
// grades and combined totals
DistinctionNo ← 0
MeritNo ← 0
PassNo ← 0
FailNo ← 0

FOR StudentCounter ← 1 to ClassSize
  TotalMark[StudentCounter] ← 0
NEXT StudentCounter

// programming techniques of iteration, selection, totalling, counting and output are used
```

Question	Answer	Marks

13	FOR StudentCounter ← 1 to ClassSize
	  FOR SubjectCounter ← 1 to SubjectNo
	  TotalMark[StudentCounter] ← TotalMark[StudentCounter] + StudentMark[StudentCounter,
	  SubjectCounter]
	  NEXT SubjectCounter
	  AverageMark[StudentCounter] ← INT((TotalMark[StudentCounter] / SubjectNo) + 0.5)
	  OUTPUT "Name ", StudentName[StudentCounter]
	  OUTPUT "Combined total mark ", TotalMark[StudentCounter]
	  OUTPUT "Average mark ", AverageMark[StudentCounter]
	  IF AverageMark[StudentCounter] >= Distinction
	  THEN
	  DistinctionNo ← DistinctionNo + 1
	  OUTPUT "Grade Distinction"
	  ELSE
	  IF AverageMark[StudentCounter] >= Merit
	  THEN
	  MeritNo ← MeritNo + 1
	  OUTPUT "Grade Merit"
	  ELSE
	  IF AverageMark[StudentCounter] >= Pass
	  THEN
	  PassNo ← PassNo + 1
	  OUTPUT "Grade Pass"
	  ELSE
	  FailNo ← FailNo + 1
	  OUTPUT "Grade Fail"
	  ENDIF
	  ENDIF
	  ENDIF
	NEXT StudentCounter
	
	OUTPUT "Number of Distinctions ", DistinctionNo
	OUTPUT "Number of Merits ", MeritNo
	OUTPUT "Number of Passes ", PassNo
	OUTPUT "Number of Fails ", FailNo	

AO2: Apply knowledge and understanding of the principles and concepts of computer science to a given context, including the analysis and design of computational or programming problems			
0	**1–3**	**4–6**	**7–9**
No creditable response	At least one programming technique has been used. *Any use of selection, iteration, counting, totalling, input and output.*	Some programming techniques used are appropriate to the problem. *More than one technique seen applied to the scenario, refer to the list of techniques needed.*	The range of programming techniques used is appropriate to the problem. *All criteria stated for the scenario have been covered by the use of appropriate programming techniques, refer to the list of techniques needed.*
	Some data has been stored but not appropriately. *Any use of variables or arrays or other language-dependent data structures, e.g. Python lists.*	Some of the data structures chosen are appropriate and store some of the data required. *More than one data structure used to store data that is required by the scenario.*	The data structures chosen are appropriate and store all the data required. *The data structures used store all the data that is required by the scenario.*

AO3: Provide solutions to problems by: • **evaluating computer systems** • **making reasoned judgements** • **presenting conclusions**			
0	**1–2**	**3–4**	**5–6**
No creditable response	Program seen without relevant comments.	Program seen with some relevant comment(s).	The program has been fully commented.
	Some identifier names used are appropriate. *Some of the data structures used have meaningful names.*	The majority of identifiers used are appropriately named. *Most of the data structures used have meaningful names.*	Suitable identifiers with names meaningful to their purpose have been used throughout. *All the data structures used have meaningful names.*
	The solution is illogical.	The solution contains parts that may be illogical.	The program is in a logical order.
	The solution is inaccurate in many places. *Solution contains few lines of code, with errors, that attempt to perform a task given in the scenario.*	The solution contains parts that are inaccurate. *Solution contains lines of code, with some errors, that logically perform tasks given in the scenario. Ignore minor syntax errors.*	The solution is accurate. *Solution logically performs all the tasks given in the scenario. Ignore minor syntax errors.*
	The solution attempts at least one of the requirements. *Solution contains lines of code that attempt at least one task given in the scenario.*	The solution meets most of the requirements. *Solution contains lines of code that perform most tasks given in the scenario.*	The solution meets all the requirements given in the question. *Solution performs all the tasks given in the scenario.*

BLANK PAGE