Cambridge Assessment International Education

Cambridge IGCSE™

COMPUTER SCIENCE **0478/02**

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

MARK SCHEME B

Maximum Mark: 75

Specimen

This document has **14** 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:

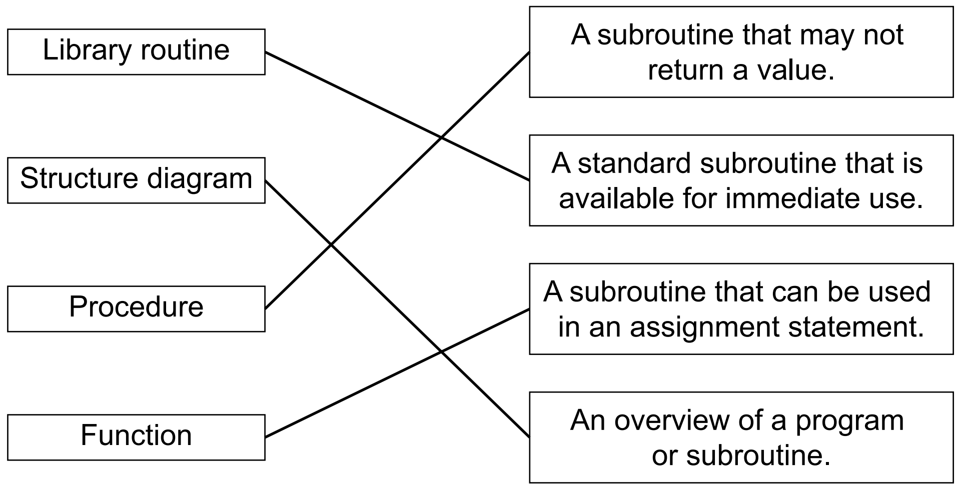
Marks awarded are based solely on the requirements as defined in the mark scheme. Marks should not be awarded with grade thresholds or grade descriptors in mind.

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

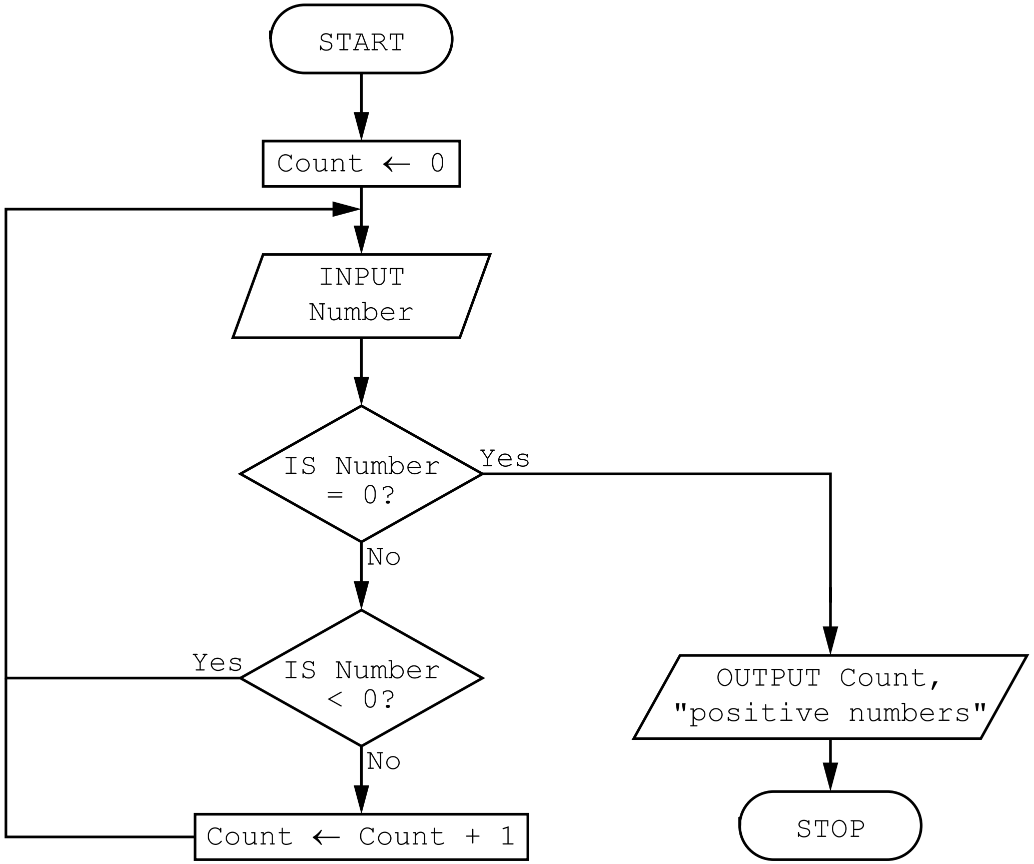
Note: No marks are awarded for using brand names of software packages or hardware.

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

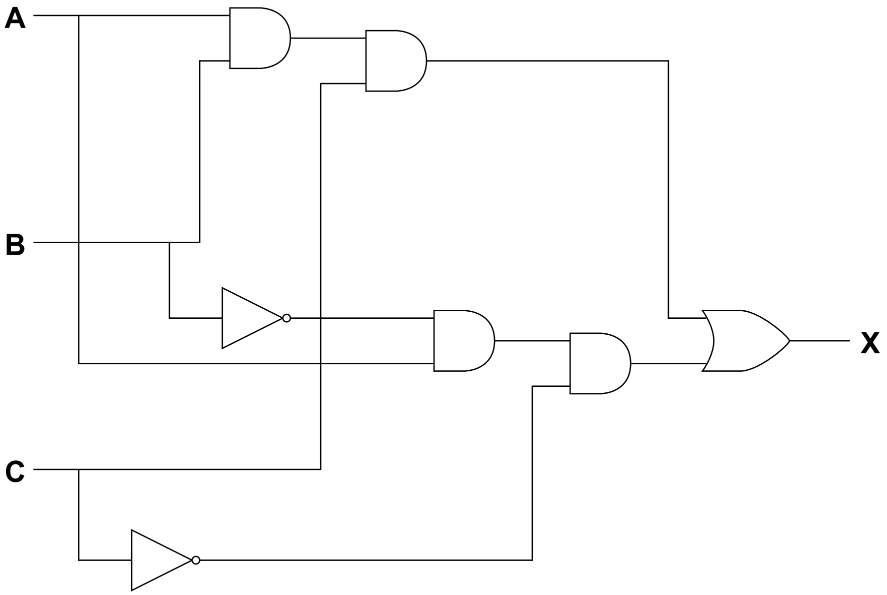
Question	Answer	Marks
2	**Programming concept** **Description**  Library routine Structure diagram Procedure Function A subroutine that may not return a value. A standard subroutine that is available for immediate use. A subroutine that can be used in an assignment statement. An overview of a program or subroutine. **One** mark for each correct line (**max 3**). 0 correct 0 marks 1 correct 1 mark 2 correct 2 marks 3 or 4 correct 3 marks Each box must have only one connection.	**3**

Question	Answer	Marks
3	1 mark for value and 1 mark for appropriate reason, e.g.: *Value 1* `2` boundary should be accepted as weight OK *Value 2* `two` erroneous/abnormal should be rejected	**4**

Question	Answer	Marks
4(a)	**One** mark for each item correctly circled • open • write	**2**
4(b)	**One** mark for each correct point **max two**: • before trying to open the file • check the file exists • if the file does not exist, then output a suitable error message.	**2**

Question	Answer	Marks
5(a)	**One** mark per correct pair of actions, processes, Input/Output, tests (apart from START and STOP) max 3. **One** mark for complete flowlines. **One** mark for working flowlines. **One** mark for correct use of flowchart symbols.  START Count ← 0 INPUT Number IS Number = 0? Yes No IS Number < 0? Yes No Count ← Count + 1 OUTPUT Count, "positive numbers" STOP	**6**
5(b)	Any **two** from: • Use another counter/variable and initialise to zero before looping • Increment this counter/variable when the number is less than zero/count all numbers and subtract the positive numbers • Output this counter/variable at the end // Output both counters at the end	**2**

Question	Answer	Marks
6	**One** mark for each error identified plus suggested correction (the corrected lines must be written in full). Correct lines: Line 4 `WHILE Number <= 99 OR Number >= 1001` Line 7 `Num[Index] ← Number` Line 9 `NEXT Index` Line 10 `PRINT Count`	**4**

Question	Answer	Marks
7(a)	**One** mark for each correct NOT gate and OR gate with correct direction of input(s), **three** marks for four correct **AND** gates **or** **two** marks for three correct **AND** gates **or** **one** mark for two correct **AND** gates with correct direction of inputs: 	**6**

Question	Answer	Marks
7(b)	X = ((A AND NOT B AND NOT C) OR (A AND B AND C)) **One** mark for each correct part of the logic expression: (A AND NOT B AND NOT C) OR (A AND B AND C)	**3**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>8</td><td>
<table>
<tr><th>Weight</th><th>Reject</th><th>TotalWeight</th><th>OUTPUT</th></tr>
<tr><td></td><td>0</td><td>0</td><td></td></tr>
<tr><td>13</td><td></td><td>13</td><td></td></tr>
<tr><td>17</td><td></td><td>30</td><td></td></tr>
<tr><td>26</td><td>1</td><td></td><td></td></tr>
<tr><td>25</td><td></td><td>55</td><td></td></tr>
<tr><td>5</td><td></td><td>60</td><td></td></tr>
<tr><td>10</td><td></td><td>70</td><td></td></tr>
<tr><td>15</td><td></td><td>85</td><td></td></tr>
<tr><td>35</td><td>2</td><td></td><td></td></tr>
<tr><td>20</td><td></td><td>105</td><td></td></tr>
<tr><td></td><td></td><td>85</td><td>Weight of items 85 Number of items rejected 2</td></tr>
<tr><td>1 mark</td><td>1 mark</td><td>1 mark to 1st 85
1 mark 105, 85</td><td>1 mark
Output must be exact</td></tr>
</table>
</td><td><b>5</b></td></tr>
</table>

Question	Answer	Marks
9	**B**	**1**

Question	Answer	Marks

10(a)	`P    Computer Science`
	`Q    16`
	`R    Science`
	`S    7`
	`T    Sci`
		5

10(b)	**One** mark correct function assigned to `F` **one** mark correct parameters `F ← SUBSTRING(P,1,8)`	**2**

Question	Answer	Marks
11(a)	*Fields* 5 *Records* 8	**2**
11(b)	Any **two** from: • length check • type check • presence check • format check	**2**
11(c)	**One mark** content and **one mark** field order: 03 Nov Acoustic Evening	**2**

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>12(a)</td><td>The whole algorithm must be rewritten for full marks.

<b>One</b> mark for each of the following:
<ul>
<li>initialising counter outside the loop</li>
<li>updating counter inside loop</li>
<li>suitable exit value at start of loop</li>
<li>correct use of <code>WHILE … DO … ENDWHILE</code></li>
</ul>
Example:
<pre>
B ← FALSE
INPUT Num
Counter ← 1
WHILE Counter <= 12 DO
   IF A[Counter] = Num
      THEN
         B ← TRUE
   ENDIF
   Counter ← Counter + 1
ENDWHILE
</pre></td><td><b>4</b></td></tr>
<tr><td>12(b)</td><td>Linear search</td><td><b>1</b></td></tr>
<tr><td>12(c)</td><td>Any <b>three</b> from:
<ul>
<li><code>WHILE</code> has criteria check at start // pre-condition</li>
<li>code inside <code>WHILE</code> may never run</li>
<li><code>REPEAT UNTIL</code> has criteria check at end // post-condition</li>
<li><code>REPEAT UNTIL</code> will always run at least once</li>
</ul></td><td><b>3</b></td></tr>
</table>

Question	Answer	Marks

13	Read the whole answer:
	Check if each requirement listed below has been met. Requirements may be met using a suitable built-in function from the programming language used (Python, VB.NET or Java)
	On script tick if requirement met, cross if no attempt seen, omission mark and/or comment if partially met (see marked scripts).
	Use the tables for A02 and A03 below to award a mark in a suitable band using a best fit approach
	Then add up the total.
	Marks are available for:
	AO2 (maximum 9 marks)
	AO3 (maximum 6 marks)
	
	Techniques required:
	R1 Procedure that takes the hospital number as a parameter (use of procedures and parameters)
	R2 Check if hospital number valid (selection, use of 1D array)
	R3 Check temperature reading (selection, use of 2D array)
	R4 Check pulse reading (selection, use of 2D array)
	R5 Output appropriate messages for each selection (output with appropriate messages)
	
	Data Structures required:
	The names underlined must be used as given in the scenario
	Arrays or lists `Patient`, `Readings`
	Variables `HospitalNumber`
	Constants `TempHigh`, `TempLow`, `PulseHigh`, `PulseLow` could be variables
	
	Example 15 mark answer in pseudocode.
	
	`//Declaration of variables and constants`
	`CONSTANT TempHigh = 37.2`
	`CONSTANT TempLow = 31.6`
	`CONSTANT PulseHigh = 100.0`
	`CONSTANT PulseLow = 55.0`
		15

Question	Answer	Marks
13		

```
PROCEDURE CheckPatient(HospitalNumber :INTEGER)
   IF HospitalNumber >=1 AND HospitalNumber <=1000 // check for valid hospital number
      THEN
         OUTPUT "Name of Patient ",Patient(HospitalNumber)
         IF Reading[HospitalNumber,1] <= TempHigh AND
            Reading[HospitalNumber,1] >= TempLow AND
            Reading[HospitalNumber,2] <= PulseHigh AND
            Reading[HospitalNumber,2] >= PulseLow // check if all readings normal
               THEN
                  OUTPUT "Normal readings"
         ENDIF
         IF (Reading[HospitalNumber,1] <= TempHigh AND
            Reading[HospitalNumber,1] >= TempLow) AND
            (Reading[HospitalNumber,2] > PulseHigh OR
            Reading[HospitalNumber,2] < PulseLow) // check if pulse out of range
               THEN
                  OUTPUT "Warning Pulse"
         ENDIF
         IF (Reading[HospitalNumber,1] > TempHigh OR
            Reading[HospitalNumber,1] < TempLow) AND
            (Reading[HospitalNumber,2] <= PulseHigh AND
            Reading[HospitalNumber,2] >= PulseLow) // check if temp out of range
            THEN
               OUTPUT "Warning temperature"
         ENDIF
         IF (Reading[HospitalNumber,1] > TempHigh OR
            Reading[HospitalNumber,1] < TempLow) AND
            (Reading[HospitalNumber,2] > PulseHigh OR
            Reading[HospitalNumber,2] < PulseLow) // check if both out of range
               THEN
                  OUTPUT "Severe warning, Pulse and temperature"
         ENDIF
      ELSE
         OUTPUT "Hospital number not valid"
   ENDIF
ENDPROCEDURE
```

<table>
<tr><th colspan="4">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</th></tr>
<tr><th>0</th><th>1–3</th><th>4–6</th><th>7–9</th></tr>
<tr><td rowspan="2">No creditable response</td><td>At least one programming technique has been used.
<em>Any use of selection, iteration, counting, totalling, input and output.</em></td><td>Some programming techniques used are appropriate to the problem.
<em>More than one technique seen applied to the scenario, refer to the list of techniques needed.</em></td><td>The range of programming techniques used is appropriate to the problem.
<em>All criteria stated for the scenario have been covered by the use of appropriate programming techniques, refer to the list of techniques needed.</em></td></tr>
<tr><td>Some data has been stored but not appropriately.
<em>Any use of variables or arrays or other language-dependent data structures, e.g. Python lists.</em></td><td>Some of the data structures chosen are appropriate and store some of the data required.
<em>More than one data structure used to store data that is required by the scenario.</em></td><td>The data structures chosen are appropriate and store all the data required.
<em>The data structures used store all the data that is required by the scenario.</em></td></tr>
</table>

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.*