GradeMax

CAMBRIDGE
International Education

# 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.



© Cambridge University Press & Assessment 2025

GradeMax



## 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.

© Cambridge University Press & Assessment 2025 

GradeMax



## 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 |
|  | Too vague |

© Cambridge University Press & Assessment 2025 

GradeMax


**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 |

© Cambridge University Press & Assessment 2025 

GradeMax

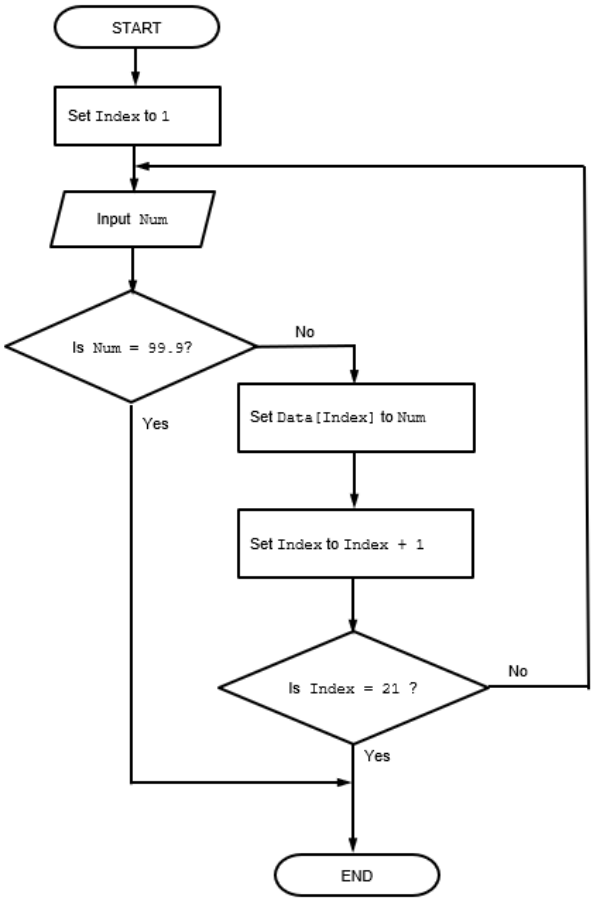


<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>1(a)</td><td>
<table>
<tr><th>Pseudocode example</th><th>Selection</th><th>Iteration</th><th>Subroutine</th></tr>
<tr><td>IF Status = FALSE THEN<br>  FOR Count ← 1 TO 20<br>  CALL Reset(Count)<br>  NEXT Count<br>ENDIF</td><td>✓</td>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

© Cambridge University Press & Assessment 2025 

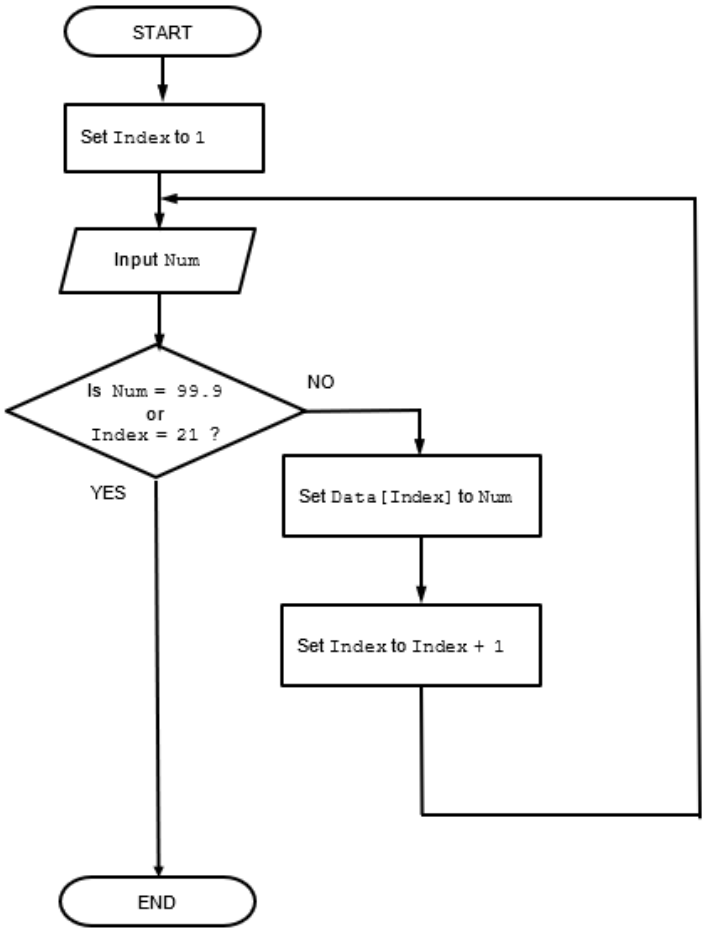
GradeMax



| Question | Answer | Marks |
|---|---|---|
| 2 | <br>Mark as follows:<br>**MP1** Initialisation of the array index<br>**MP2** Input number **in a loop**<br>**MP3** Test for termination value (99.9) **and** correct ending<br>**MP4** Test for array full / maximum 20 iterations **and** correct ending<br>**MP5** Assign number to `Data` array element **and** increment index | **5** |

GradeMax



| Question | Answer | Marks |
|---|---|---|
| 2 | Alternative solution:   | |

GradeMax



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

© Cambridge University Press & Assessment 2025 

GradeMax



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

© Cambridge University Press & Assessment 2025 

GradeMax



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

© Cambridge University Press & Assessment 2025 

GradeMax



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

© Cambridge University Press & Assessment 2025 

GradeMax



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

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

GradeMax



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

© Cambridge University Press & Assessment 2025 

GradeMax



| Question | Answer | Marks |
|---|---|---|
| 8(c) | **MP1** Save to/Open a **(text) file**<br>**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<br>**MP1** Store the **return date** of each loan // the **start date <u>and</u>** the **loan period**<br>**MP2** Algorithm will do a **calculation** involving the MP1 data item(s)<br><br>OR<br><br>The student email address solution<br>**MP3** Store the student <u>email address</u><br>**MP4** In order **to send** the email // the student is alerted to return the book<br><br>**Max 2** | **2** |

© Cambridge University Press & Assessment 2025 