Cambridge Assessment International Education

# Cambridge IGCSE™

**COMPUTER SCIENCE** **0478/21**

Paper 2 Problem-solving and Programming **October/November 2022**

MARK SCHEME

Maximum Mark: 50

**Published**

This document consists of **9** printed pages.

**[Turn over**

## 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:<br><br>Marks must be awarded in line with:<br><br>• the specific content of the mark scheme or the generic level descriptors for the question<br>• the specific skills defined in the mark scheme or in the generic level descriptors for the question<br>• the standard of response required by a candidate as exemplified by the standardisation scripts. |
| GENERIC MARKING PRINCIPLE 2:<br><br>Marks awarded are always **whole marks** (not half marks, or other fractions). |
| GENERIC MARKING PRINCIPLE 3:<br><br>Marks must be awarded **positively**:<br><br>• 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<br>• marks are awarded when candidates clearly demonstrate what they know and can do<br>• marks are not deducted for errors<br>• marks are not deducted for omissions<br>• 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:<br><br>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.

**Please note the following further points:**

The words in **bold** in the mark scheme are important text that needs to be present, or some notion of it needs to be present. It does not have to be the exact word, but something close to the meaning.

If a word is underlined, this **exact** word must be present.

A single forward slash means this is an alternative word. A double forward slash means that this is an alternative mark point.

Ellipsis (…) on the end of one-mark point and the start of the next means that the candidate **cannot** get the second mark point without being awarded the first one. If a mark point has an ellipsis at the beginning, but there is no ellipsis on the mark point before it, then this is just a follow-on sentence and **can** be awarded **without** the previous mark point.

| Question | Answer | Marks |
|---|---|---|
| | **Section A** | |
| 1(a)(i) | Many correct answers, the names used must be meaningful. The names given are examples only.<br><br>**One** mark per mark point, max **five**<br><br>MP1 name one e.g. Name, Age, Gender, Type, TeamMember, AnnualFee, Paid<br>MP2 one appropriate data type must match name e.g. string, integer, char, string, Boolean, real, Boolean<br>MP3 sample data for one appropriate array<br>MP4 all arrays mentioned have appropriate names<br>MP5 all arrays mentioned have appropriate data types<br>MP6 sample data for all arrays e.g. Sue, 9, F, Junior, True, 9.00, False | **5** |
| 1(a)(ii) | **One** mark per mark point, max **two**<br><br>MP1 use a loop e.g. `FOR…NEXT` / `REPEAT…UNTIL` / `WHILE…DO`<br>MP2 to input the values for each element in every array<br>MP3 append each value input to the end of the array | **2** |
| 1(b) | **One** mark per mark point, max **two**<br><br>MP1 use a loop/ `FOR`/`REPEAT`/`WHILE` to check all members<br>MP2 use a conditional statement / `IF` to see if the paid field is marked as False<br>MP3 only transfer the data of those members whose paid field is marked as True to a set of new arrays // delete the details of the members who have not paid | **2** |

| Question | Answer | Marks |
|---|---|---|
| 1(c) | **One** mark per mark point, max **six**<br><br>MP1 loop through all the members<br>MP2 any suitable correct comparison of age<br>MP3 if 18 Change membership type to adult<br>MP4 if 50 Change membership type to senior<br>MP5 if 80 Change membership type to golden<br>MP6 check if team member<br>MP7 no store appropriate full fee<br>MP8 yes store 90% of appropriate full fee<br><br>**Example**<br>`FOR Member ← 1 TO Total`<br>`  CASE Age[Member] OF`<br>`    18 : Type[Member] ← "Adult"`<br>`    50 : Type[Member] ← "Senior"`<br>`    80 : Type[Member] ← "Golden"`<br>`  ENDCASE`<br><br>`  IF Team[Member]`<br>`    THEN`<br>`      CASE Type[Member] OF`<br>`        "Junior" : Fee[Member] ← JuniorFee * 0.9`<br>`        "Adult"  : Fee[Member] ← AdultFee * 0.9`<br>`        "Senior" : Fee[Member] ← SeniorFee * 0.9`<br>`      ENDCASE`<br>`    ELSE`<br>`      CASE Type[Member] OF`<br>`        "Junior" : Fee[Member] ← JuniorFee`<br>`        "Adult"  : Fee[Member] ← AdultFee`<br>`        "Senior" : Fee[Member] ← SeniorFee`<br>`        "Golden" : Fee[Member] ← GoldenFee`<br>`      ENDCASE`<br>`  ENDIF`<br>`NEXT Member` | 6 |

| Question | Answer | Marks |
|---|---|---|
| 1(d) | Explanation<br><br>**One** mark per mark point, max **five**<br><br>MP1 how the program checked all the members // use of loop<br>MP2 how the program identified a member who has not paid their (annual) fee // use of condition<br>MP3 how the program kept a running total of members who have not paid their (annual) fees<br>MP4 for each of the three types of membership that require a fee // excluding golden members<br>MP5 how the program calculated a percentage for members who had not paid…<br>MP6 for each of the three types of membership that require a fee<br>MP7 how the program displayed a percentage for members who had not paid<br>MP8 all three percentages displayed with suitable messages<br><br>Programming statements when used must be explained. | **5** |

| Question | Answer | Marks |
|---|---|---|
| | **Section B** | |
| 2(a) | **One** mark per mark point, max **six**<br><br>• Line 1 `100`<br>• Line 7 `Value > 100 // Value >= 101`<br>• Line 11 `Reading[Value] + 1`<br>• Line 14 `INPUT Value`<br>• Line 18 `Reading[Count]`<br>• Line 19 `Count – 1` | **6** |

| Question | Answer | Marks |
|---|---|---|
| 2(b) | **One** mark per mark point, max **three**<br><br>• use an IF/conditional statement<br>• to check if Reading[Count] not equal to zero<br>• before outputting the value // between statements 17 and 18 // code sample showing position<br><br>IF Reading[Count] <> 0<br>  THEN<br>    OUTPUT<br>ENDIF | **3** |

| Question | Answer | Marks |
|---|---|---|
| 3(a) | **Two** marks per check, description must match name of check if given, max **six**<br><br>• Check 1 use a type check<br>○ to ensure that the value is a number / integer<br>• Check 2 use a length check<br>○ to ensure that there are only 4 characters / digits<br>• Check 3 use a range check<br>○ to ensure that the value is **>= 1000 and <=9999** | **6** |
| 3(b) | **One** mark per mark point, max **three**<br><br>MP1 input the new PIN<br>MP2 input the new PIN again // ask the user to check the number on screen<br>MP3 check that both PINs are the same // confirm that it is the PIN to use<br>MP4 check that the new PIN is not the same as the old PIN | **3** |

| Question | Answer | Marks |
|---|---|---|
| 4(a) | **One** mark for each correct column, max **four** | **4** |

| `Sold` | `Stock` | `Total` | **OUTPUT** |
|---|---|---|---|
| | 50 | 0 | |
| 24 | 26 | 24 | |
| 12 | 14 | | Add new stock |
| | 64 | 36 | |
| 6 | 58 | 42 | |
| 30 | 28 | 72 | |
| 12 | 16 | | Add new stock |
| | 66 | 84 | |
| 18 | 48 | 102 | |
| -1 | | | 102 |

| Question | Answer | Marks |
|---|---|---|
| 4(b) | **One** mark for identification of error, max **one** | **3** |

- the stock level will fall below zero / become negative

**One** mark per mark point, max **two**

- before subtracting the amount Sold from Stock
- test that the stock level / Stock is greater than the rolls to be sold / Sold
- provide a suitable error message / ask to re-input

| Question | Answer | Marks |
|---|---|---|
| 5(a) | **One** mark per mark point, max **two**<br><br>• ItemCode<br>• uniquely identifies each item | **2** |
| 5(b) | **One** mark per mark point, max **three**<br><br>• correct rows Field, Table and Sort<br>• correct row Show<br>• correct Criteria row <10 or <=9 | **3** |

| | | | | |
|---|---|---|---|---|
| Field: | ItemCode | Manufacturer | Level | |
| Table: | WAREHOUSE | WAREHOUSE | WAREHOUSE | |
| Sort: | | | | |
| Show: | ☑ | ☑ | ☐ | ☐ |
| Criteria: | | | <10 | |
| or: | | | | |