Cambridge Assessment International Education

# Cambridge O Level

**COMPUTER SCIENCE** **2210/21**

Paper 2 Problem Solving and Programming **May/June 2022**

MARK SCHEME

Maximum Mark: 50

**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 May/June 2022 series for most Cambridge IGCSE, Cambridge International A and AS Level and Cambridge Pre-U components, and some Cambridge O Level components.

This document consists of **13** printed pages.

© UCLES 2022

**PUBLISHED**

## 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.

© UCLES 2022

| |
|---|
| GENERIC MARKING PRINCIPLE 5:<br><br>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:<br><br>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. |

© UCLES 2022

**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.

© UCLES 2022

| Question | Answer | Marks |
|---|---|---|
| | **Section A** | |
| 1(a) | **One** mark per mark point, max **five**<br><br>Data Structure(s), max **two**<br>MP1 arrays<br>MP2 variable(s) / constant(s)<br>Further description, max **three**<br>MP3 name(s) one or more<br>MP4 sample data for appropriate arrays or variables<br>MP5 use(s) one or more<br>Additional data structure description using the **same** data structure type, max **one**<br>MP6 two or more full descriptions of the data structure including name, sample data and use<br><br>For example:<br>An array (1) named FirstName (1) to store the first names of the members (1) such as James (1). A variable (1) could also be used to enter whether or not they wish to volunteer. (5 marks)<br><br>A variable (1) named FirstName (1) to input the first names of the members (1) such as James (1). A variable could also be used to enter whether or not they wish to volunteer, with sample data of 'yes' (1). (5 marks)<br><br>**Task 1** – becoming a member of Friends of Seaview Pier<br>Set up a system to enable people to become members of Friends of Seaview Pier and for each new member enter:<br>• their first name and last name<br>• whether or **not** they wish to work as a volunteer<br>– if they choose to volunteer, identify the area from:<br>o the pier entrance gate<br>o the gift shop<br>o painting and decorating<br>• the date of joining<br>• whether or **not** they have paid the $75 fee.<br>All of this information needs to be stored using suitable data structures. | **5** |

| Question | Answer | Marks |
|---|---|---|
| 1(b) | Explanation of how each was done. Code is allowed, but must be fully explained.<br><br>**One** mark per mark point, max **three**<br><br>MP1 add/use a (new) variable/array to total the membership fee money // total number of members who have paid<br>MP2 initialise the totalling variable to 0 at the start of the program<br>MP3 check whether the new member has paid the $75 fee using a conditional statement<br>MP4 …if they have paid, and the amount paid is being totalled, add 75 to the current running total of the membership fee total<br>MP5 …if they have paid, and the number of paid members is being totalled, add 1 to the total number of members who have paid<br>MP6 …if they have paid, and the number of paid members is being totalled, multiply total by 75, to give total paid. | **3** |
| 1(c) | Any code must include a description of what it is for.<br><br>**One** mark per mark point, max **three**<br>Max **two** for just naming different appropriate validation checks<br><br>MP1 apply a presence check // to ensure that data has been entered (to the question do you want to work as a volunteer?)<br>MP2 apply a type check // to ensure that data has been entered of the correct data type e.g. integer if the response required is 1 or 0<br>MP3 checking the valid inputs would be either yes or no // apply a check to ensure that the data matches the expected input<br>MP4 the valid inputs would be to choose in which area the new member wishes to volunteer e.g. a list of areas<br>MP5 if input is not valid, an error message is output (and a new input is requested)<br>MP6 if input is valid, the program continues | **3** |

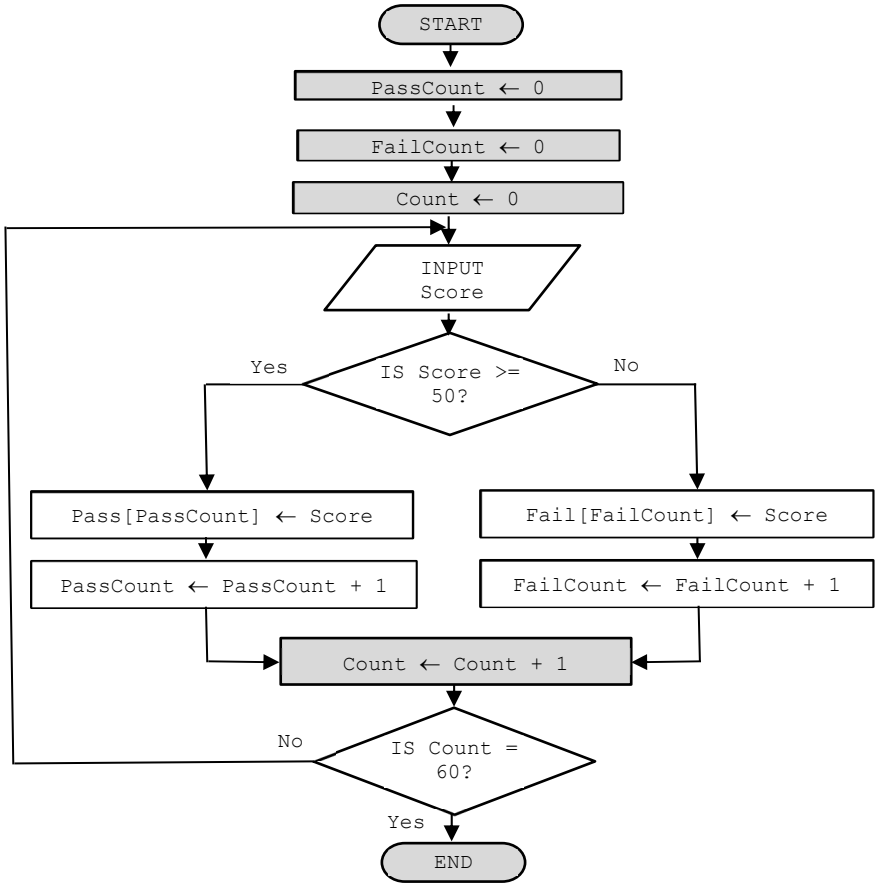
© UCLES 2022

| Question | Answer | Marks |
|---|---|---|
| 1(d) | **One** mark per mark point, max **five**<br>MP1 input for sponsor's name / name taken from previous data // input for message on plaque<br>MP2 both inputs correct with appropriate prompts // input for message on plaque correct with appropriate prompt and name taken from previous data<br>MP3 output of input(s) for confirmation …<br>MP4 … method for sponsor to confirm that the input(s) are correct<br>MP5 method to enable re-entry of message on plaque if errors made<br>MP6 charge of $200 acknowledged / charged / confirmed / displayed<br>MP7 (name and) message stored in arrays<br>MP8 array index incremented for next sponsor<br><br>**Task 3** – sponsoring a wooden plank<br>Add an additional option to the program in **Task 1** to enable the pier's wooden planks to be sponsored. Separate data structures should be used to store the names of the individuals and the short messages they would like to have written on their brass plaque. An output would display everything that was input for the sponsor to confirm. If errors are found, the program should allow data to be re-entered. Once complete, the data is stored and the sponsor is charged $200.<br><br>**Example answer**<br>`ArrayIndex ← 0 //initial array's index`<br>`Check ← "N"`<br>`WHILE Check <> "Y"`<br>`  OUTPUT "Enter your name"`<br>`  INPUT Name`<br>`  OUTPUT "Enter the message you would like on your brass plaque"`<br>`  INPUT Message`<br>`  OUTPUT "The data you have entered is, name: ", Name, " and your`<br>`  message for the plaque is: ", Message`<br>`  OUTPUT "Is this correct (Y or N)"`<br>`  INPUT Check`<br>`  IF Check <> "Y"`<br>`  THEN`<br>`  OUTPUT "The data entered is incorrect, please re-enter"`<br>`  ENDIF`<br>`ENDWHILE` | **5** |

| Question | Answer | Marks |
|---|---|---|
| 1(d) | `PlankName[ArrayIndex] ← Name`<br>`Message[ArrayIndex] ← Message`<br>`ArrayIndex ← ArrayIndex + 1`<br>`OUTPUT "The fee for this service is $200"`<br>`//some method of paying the fee or acknowledgement of the fee` | |
| 1(e) | Explanation of how each was done. Code is allowed, but must be fully explained.<br><br>**One** mark per mark point, max **four**<br><br>MP1 a menu is provided/options are displayed so that the user can choose which of the lists they wish to see<br>MP2 the user inputs a number/code as shown on the menu corresponding to their menu choice<br>MP3 attempt to validate input<br>MP4 … if it does not match an option, give an error message and ask for re-input<br>MP5 … if it matches an option, a range of IF statements/conditional statements/CASE statement are/is used to compare the input with the available options<br>MP6 … output the chosen list e.g. using a loop to output the contents of the appropriate first and second name arrays corresponding to the user input<br>MP7 identification of empty list and appropriate action<br><br>**Task 2** – using the membership data<br>Extend the program in **Task 1** so that a list of the first and last names of members can be output in any of the following categories:<br>• Members who have chosen to work as volunteers.<br>• Volunteers who would like to work at the pier entrance gate.<br>• Volunteers who would like to work in the gift shop.<br>• Volunteers who would like to help with painting and decorating tasks.<br>• Members whose membership has expired (they have **not** re-joined this year).<br>• Members who have **not** yet paid their $75 fee. | **4** |

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td colspan="3">Section B</td></tr>
<tr><td>2</td><td>
Four marks for five correct rows<br>
Three marks for four correct rows<br>
Two marks for three correct rows<br>
One marks for two correct rows<br><br>
<table>
<tr><th rowspan="2">Description</th><th colspan="5">Data type</th></tr>
<tr><th>Boolean</th><th>Char</th><th>Integer</th><th>Real</th><th>String</th></tr>
<tr><td>a single character from the keyboard</td><td></td><td>✓</td><td></td><td></td><td></td></tr>
<tr><td>multiple characters from the keyboard</td><td></td><td></td><td></td><td></td><td>✓</td></tr>
<tr><td>only one of two possible values</td><td>✓</td><td></td><td></td><td></td><td></td></tr>
<tr><td>only whole numbers</td><td></td><td></td><td>✓</td><td></td><td></td></tr>
<tr><td>any number</td><td></td><td></td><td></td><td>✓</td><td></td></tr>
</table>
</td><td>4</td></tr>
</table>

| Question | Answer | Marks |
|---|---|---|
| 3 | **One** mark per mark point, max **four**<br><br>• Normal test data computerscience@cambridge.org.uk<br>• Reason this is a valid email address (containing the @ symbol) and should be accepted<br><br>• Erroneous test data computerscienceisgreat<br>• Reason this is just a string, and should be rejected (as an email address needs a single '@') | **4** |

| Question | Answer | Marks |
|---|---|---|
| 4(a) | **One** mark per mark point, max **six**<br><br>MP1 input box<br>MP2 correct check of `Score`<br>MP3 assign `Score` to `Pass` correctly<br>MP4 assign `Score` to `Fail` correctly<br>MP5 increment **both** array counters<br>MP6 correct check of number of scores<br><br><br> | **6** |

| Question | Answer | Marks |
|---|---|---|
| 4(b) | **One** mark per mark point, max **four**<br><br>MP1 appropriate conditional loop structure<br>MP2 correct identification of invalid input<br>MP3 appropriate error message<br>MP4 repeated input of score until correct<br><br>`WHILE Score < 0 OR Score > 100 (DO)`<br>`  OUTPUT "Your entry must be between 0 and 100, inclusive, please try again "`<br>`  INPUT Score`<br>`ENDWHILE`<br><br>Or:<br><br>`REPEAT`<br>`  IF Score < 0 OR Score > 100`<br>`    THEN`<br>`      OUTPUT "Your entry must be between 0 and 100, inclusive, please try again "`<br>`      INPUT Score`<br>`  ENDIF`<br>`UNTIL Score >= 0 AND Score <= 100` | **4** |

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>5(a)</td><td>

**One** mark per mark point, max **five**

MP1 correct `Counter` and `Limit` columns
MP2 correct `Value` column
MP3 correct `First` column
MP4 correct `Last` column
MP5 correct OUTPUT

<table>
<tr><th>Counter</th><th>Value</th><th>First</th><th>Last</th><th>Limit</th><th>OUTPUT</th></tr>
<tr><td></td><td></td><td>0</td><td>0</td><td>8</td><td></td></tr>
<tr><td>1</td><td>66</td><td></td><td></td><td></td><td></td></tr>
<tr><td>2</td><td>606</td><td>6</td><td>6</td><td></td><td>606</td></tr>
<tr><td>3</td><td>6226</td><td></td><td></td><td></td><td></td></tr>
<tr><td>4</td><td>8448</td><td></td><td></td><td></td><td></td></tr>
<tr><td>5</td><td>642</td><td>6</td><td>2</td><td></td><td></td></tr>
<tr><td>6</td><td>747</td><td>7</td><td>7</td><td></td><td>747</td></tr>
<tr><td>7</td><td>77</td><td></td><td></td><td></td><td></td></tr>
<tr><td>8</td><td>121</td><td>1</td><td>1</td><td></td><td>121</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td></td></tr>
</table>

</td><td>**5**</td></tr>
</table>

© UCLES 2022

| Question | Answer | Marks |
|---|---|---|
| 5(b) | **One** mark per mark point, max **two**<br>• checks for / outputs 3-digit numbers<br>• … where the first and last digit are the same | **2** |

| Question | Answer | Marks |
|---|---|---|
| 6(a) | 8 | **1** |
| 6(b) | The primary key field must be unique/different for each record in the table | **1** |
| 6(c) | **One** mark per mark point, max **three**<br><br>• correct fields and table named correctly<br>• correct sort and show box rows<br>• correct search criteria | **3** |

| | | | | | |
|---|---|---|---|---|---|
| Field: | GameID | GameName | GamePrice | NumberStock | OnOrder |
| Table: | GAMES | GAMES | GAMES | GAMES | GAMES |
| Sort: | | Ascending | | | |
| Show: | ☑ | ☑ | ☑ | ☐ | ☐ |
| Criteria: | | | | =0 | ="Y" |
| or: | | | | | |

© UCLES 2022