|  |  |
| --- | --- |
| Y:\Operations\Document_Production\Documents\Mark Schemes\Resources\logos\IGCSE.eps | **Cambridge Assessment International Education** Cambridge International General Certificate of Secondary Education |

Computer SCIENCE **0478/22**

Paper 2   **March 2021**

MARK SCHEME

Maximum Mark: 50

**IMPORTANT NOTICE**

Mark Schemes have been issued on the basis of **one** copy per Assistant examiner and **two** copies per Team

Leader.

|  |
| --- |
| **Final Publication** |

**Cambridge Assessment International Education – 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. |

| **Question** | **Answer** | | | **Marks** |
| --- | --- | --- | --- | --- |
|  | **Section A** | | |  |
| 1(a) | Variable answer (example only)  Data type integer/int  Validation two distinct different points **OR** one point and an expansion  **One** mark per mark point, max **two**   * restricted input to positive numbers * checked if input was numeric * ensured that input was a whole number / integer * checked if a value was input   **Or** one point and an expansion **two** marks, for example   * restricted input to positive numbers (1) by using a range/limit check / REPEAT… UNTIL looping until value greater than zero (1) * checked if input was numeric (1) by using a type check (1) * ensured the value input was a whole number (1) any suitable method e.g. rounding to whole number * checked if a value was input (1) by using a presence check (1) | | | **4** |
| 1 (b) | Variable noAttempts (example only)  Data type integer/int  Use storing the number of attempts a pupil has made on a question | | | **3** |
| 1(c) | | **One** mark per mark point, max **six**   1. initialisation (must include number of correct answers) 2. input name and table number with prompts 3. validate table number **and reinput** 4. ensure each question uses a different multiplier 5. display question 6. input **and validate** answer 7. check if answer correct 8. update score appropriately 9. repeat for 5 questions 10. display score **and name** at end … 11. …with an attempt to display an appropriate message dependent on score   Example  Score = 0  Multiplier[1] ← 2  Multiplier[2] ← 5  Multiplier[3] ← 7  Multiplier[4] ← 8  Multiplier[5] ← 11  OUTPUT "Multiplication Table Test"  OUTPUT "Please enter your name"  INPUT Name  REPEAT  OUTPUT "Please enter the table (2 to 12) you would like to be tested"  INPUT Table  UNTIL (Table >=2) AND (Table <=12)  FOR Count = 1 TO 5  OUTPUT ("Question ", Count)  OUTPUT (Multiplier[Count], " X ", Table, " = ")  REPEAT  INPUT Answer  UNTIL Answer > 0 AND Answer = INT(Answer)  IF Answer = Multiplier[Count] \* Table  THEN  Score ← Score + 1  ENDIF  NEXT  OUTPUT (Name, " your score is ", Score)  IF Score = 5  THEN  OUTPUT ("Well done full marks")  ELSE  OUTPUT ("Have another practice")  ENDIF | **6** | |
| 1(d) | | Explanation  **One** mark per mark point illustrated with a suitable programming statement, max **four**   1. extra prompt and input to choose the number of questions and/or mixed set 2. method of choosing number of questions and/or mixed set 3. extra prompt and input for number of questions 4. explanation of validation for either input 5. how the end value for the question loop was changed if required 6. how the program was changed to allow for mixed tables 7. … how the program ensured that the test used more than one multiplication table | **4** | |
| 1 (e) | | Explanation  **One** mark per mark point, max **three**   1. how to provide 3 answers, including the correct one 2. how to ensure that all 3 answers are different 3. how 3 answers are displayed e.g. output all 3 answers as a numbered list 4. how to select the correct answer e.g. input the number of the correct answer 5. how to ensure that the correct answer is not always in the same position | **3** | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Section B** | |  |
| 2 (a) | * 12 * 13 * 07 * 15 | | **4** |
| 2(b) | | * 27 (example many correct answers any whole number 1-99 inclusive) * 106 (example many correct answers) * 99/1 | **3** |
| 2(c) | | **One** mark per mark point, max **four**   * **new** variable for minimum… * … set to first value/high value * … at start of program / before line 4 * test input / D for less than minimum * … replace value minimum if input less than * … after line 7 and before line 14 * new output for minimum (with appropriate message) * … at end of program // after line 14 | **4** |
| 3 | **Pseudocode statement Flowchart symbol**  IF X > 12  INPUT X  X ← Y + Z  OUTPUT X  **One** mark for a single line to decision box from IF X > 12  **One** mark for a single line to process box from X ← Y + Z  **Two** marks for two single lines from INPUT X and OUTPUT X to output box  **OR**  **One** mark for a single line from INPUT X or OUTPUT X to output box // two single lines from INPUT X and OUTPUT X to process box | | **4** |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4(a) | One mark for each correct column   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Bag Weight** | **Accept** | **Over** | **Under** | **Error** | **Total** | **OUTPUT** | |  | 0 | 0 | 0 |  |  | (Enter weight of first cookie bag) | | 1.05 | 1 |  |  | 0 |  | (Weight of next bag?) | | 0.99 | 2 |  |  | 0 |  | (Weight of next bag?) | | 1.2 |  | 1 |  | 1 |  | (Weight of next bag?) | | 0.85 |  |  | 1 | 2 |  | (Weight of next bag?) | | 1.1 | 3 |  |  | 0 |  | (Weight of next bag?) | | 0.9 | 4 |  |  | 0 |  | (Weight of next bag?) | | 1.5 |  | 2 |  | 1 |  | (Weight of next bag?) | | 0.95 | 5 |  |  | 0 |  | (Weight of next bag?) | | 1.05 | 6 |  |  | 0 |  | (Weight of next bag?) | | 1.00 | 7 |  |  | 0 |  | (Weight of next bag?) | | 1.07 | 8 |  |  | 0 |  | (Weight of next bag?) | | 0.89 |  |  | 2 | 2 |  | (Weight of next bag?) | | -10 |  |  |  |  | 4 |  | |  |  |  |  |  |  | Number of bags weighed 4 | |  |  |  |  |  |  | Number overweight 2 | |  |  |  |  |  |  | Number underweight 2 | |  |  |  |  |  |  |  | | **7** |
| 4(b) | **Error** Total ← Accept – Over – Under // line 26  **Correction** Total ← Accept + Over + Under  Condone = for ← | **2** |

|  |  |  |
| --- | --- | --- |
| 5(a) | **One** mark for data type and reason  SIZE text, expressed as a single word // Boolean, only two choices  PRICE currency, needs to be expressed as Rupees/money  NUMBERSOLD number, integer values/could be used in calculations | **3** |
| 5(b) | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Field: | SIZE | FILLING | PRICE | NUMBERSOLD | | Table: | CHOCBAR | CHOCBAR | CHOCBAR | CHOCBAR | | Sort: |  |  |  |  | | Show: | 🞏 | 🗹 | 🗹 | 🗹 | | Criteria: | = "small" |  |  | <10 | | or: |  |  |  |  |   **One** mark for correct rows Field, Table and Sort  **One** mark for correct Show row  **One** mark for correct Criteria row | **3** |