|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **1(a).** | The following table contains several definitions of terms that are used in Computer Science.     |  |  | | --- | --- | | **Letter** | **Definition** | | **A** | Cleaning up data entered by removing non-standard characters | | **B** | Hiding or removing irrelevant details from a problem to reduce complexity | | **C** | Checking that the user is allowed to access the program | | **D** | Breaking a complex problem down into smaller problems | | **E** | Repeating elements of a program | | **F** | Converting one data type to another, for example converting an integer to a real number |   Write the letter of the definition that matches each keyword in each space.     |  |  |  | | --- | --- | --- | |  | Decomposition | .................. | |  | Abstraction | .................. | |  | Input sanitisation | .................. | |  | Casting | .................. | | **[4]** | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(b).** | 1. Write a pseudocode statement to assign the value 7.3 to a variable with the identifier timer     **[1]**   1. State the most appropriate data type for the variable timer.     **[1]** | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **2(a).** | Dru writes the following program using a high-level language.     |  |  |  | | --- | --- | --- | |  | 01 | function newscore(a,b) | |  | 02 | temp = a\*b | |  | 03 | temp = temp + 1 | |  | 04 | return temp | |  | 05 | endfunction | |  | 06 | score = 18 | |  | 07 | name = "Dru" | |  | 08 | print (score) | |  | 09 | print ("name") | |  | 10 | print (newscore(score,2)) | |  | 11 | print (score) |   The following table contains the program code for each line where this program outputs values.  State the values output by the program on each of the lines.     |  |  |  | | --- | --- | --- | | **Line** | **Program code** | **Value output** | | 08 | print (score) |  | | 09 | print ("name") |  | | 10 | print (newscore(score,2)) |  | | 11 | print (score) |  |      |  | | --- | | **[4]** | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(b).** | Describe the advantages of writing the program in a high-level language instead of in assembly language.        **[2]** | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(c).** | Describe how a character set is used to represent the string value stored in the variable name        **[2]** | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **3(a).** | A vending machine has the following options available.     |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | |  |  |  | | --- | --- | --- | | **Item code** | **Item name** | **Price** | | A1 | Crisps, bacon flavour | £0.75 | | A2 | Crisps, salted | £0.75 | | B1 | Chocolate bar | £0.90 | | C1 | Apple pieces | £0.50 | | C2 | Raisins | £0.85 | |  |   Users insert coins into the vending machine and then enter the two character item code of their selection. If the user has inserted enough money, the vending machine will release the chosen item and output any change required. If the user enters an invalid item code then a suitable error message is displayed.  The vending machine is tested before it is released.   1. Explain the purpose of testing the vending machine.         **[2]**   1. Describe the difference between iterative testing and final testing.         **[2]**   1. Complete the following test plan for the vending machine.      |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | |  |  |  | | --- | --- | --- | | **Code entered** | **Money inserted** | **Expected result** | | B1 | £1 | Chocolate bar served, £0.10 change given | |  | £0.85 | Raisins served, no change given | | C1 |  | Error – not enough money inserted | | C3 | £0.75 |  | | | **[3]** | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(b).** | The algorithm for one section of a vending machine program is shown in pseudocode.  if money >= price then       venditem()       giveChange(money – price)   else       print("Error – not enough money inserted")   endif   1. Give the identifier of **one** variable used in the algorithm.   **[1]**   1. State how many parameters are passed into the giveChange() subroutine.   **[1]** | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(c).** | Draw the vending machine algorithm in the part above as a flowchart.     |  | | --- | |  |      |  | | --- | | **[5]** | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(d).** | When writing the program for the vending machine, maintainability was considered.   1. Identify **two** ways that the program in the part above has been made more maintainable.      |  |  |  | | --- | --- | --- | |  | 1 |  | |  |  | | |  | 2 |  | |  |  | | | **[2]** | | |  1. Give **one** additional way that the maintainability of the program can be improved.     **[1]** | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(e).** | The vending machine stores the quantity of items available in a database table called ITEMS. The current contents of ITEMS is shown:     |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | |  |  |  | | --- | --- | --- | | **ItemCode** | **ItemName** | **Stock** | | A1 | Crisps, bacon flavour | 6 | | A2 | Crisps, salted | 2 | | B1 | Chocolate bar | 12 | | C1 | Apple pieces | 18 | | C2 | Raisins | 7 | |  |   Complete the following SQL statement to display the item code for all items that have fewer than 10 in stock.     |  |  | | --- | --- | | **SELECT** |  |      |  |  | | --- | --- | | **FROM** |  | |  | | | **[4]** | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(f).** | The vending machine can be in one of three states: on, off or suspended. A user can change the state of the vending machine by using the following algorithm.      newstate = input("Enter the new state : ")       switch newstate:          case "on":             statevalue = 1          case "off":             statevalue = 2          case "suspended":             statevalue = 3          default:             print("Invalid state")       endswitch   Rewrite the algorithm to perform the same actions using IF statements in place of the switch statement.                **[5]** | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **4(a).** | Convert the binary value **1110 0011** into hexadecimal.        **[2]** | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(b).** | Convert the denary value **105** into an 8 bit binary number.        **[2]** | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(c).** | Give **two** reasons why computer scientists use hexadecimal to represent numbers instead of binary.     |  |  | | --- | --- | | 1 |  | |  | | | 2 |  | |  | | | **[2]** | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(d).** | DIV and MOD are both operators used in computing-related mathematics.   1. State the value of 13 DIV 4     **[1]**   1. State the value of 13 MOD 4     **[1]** | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(e).** | Show the outcome of a right shift of three places on the binary value 0111 1000    **[1]** | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(f).** | 1. Draw the logic diagram for the logic system **P = A OR (B AND C)**      |  | | --- | |  |      |  | | --- | | **[3]** |  1. Complete the truth table for the logic system **P = NOT (A OR B)**      |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | |  |  |  | | --- | --- | --- | | **A** | **B** | **P** | | 0 | 0 | 1 | | 0 | 1 |  | | 1 | 0 |  | |  |  |  | |  | | **[4]** | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **5(a).** | The following logo is stored as a bitmap image. Each box represents one pixel, with three different colours being used in the image.     |  | | --- | |  |   State what is meant by the term image resolution.    **[1]** | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(b).** | Calculate the fewest number of bits that could be used to store the logo as a bitmap image. You must show your working.              **[4]** | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(c).** | Give **two** ways that the file size of the image could be reduced.     |  |  | | --- | --- | | 1 |  | |  | | | 2 |  | |  | | | **[2]** | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(d).** | Metadata is sometimes stored alongside images.   1. State what is meant by the term metadata.     **[1]**   1. Give **one** example of metadata that could be stored alongside the logo.     **[1]** | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **6(a).** | The following names of students are stored in an array with the identifier studentnames.  studentnames = ["Rob", "Anna", "Huw", "Emma", "Patrice", "Iqbal"]  Describe the steps that a linear search would take to find Anna in studentnames                **[4]** | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(b).** | The names of students are sorted into ascending alphabetical order using an insertion sort.  Complete the following diagram to show the stages an insertion sort would take to complete this task.  Each row represents one pass of the insertion sort algorithm. You may not need to use all empty rows.     |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **Rob** | **Anna** | **Huw** | **Emma** | **Patrice** | **Iqbal** |      |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |      |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |      |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |      |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |      |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |      |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | |  |  |  |  |  |  |      |  | | --- | | **[5]** | | |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  |  | | **(c).** | A school uses the array to call an attendance register every morning.  Write an algorithm using iteration to:     |  |  | | --- | --- | | • | display the name of each student one at a time from studentnames | | • | take as input whether that student is present or absent | | • | display the total number of present students and number of absent students in a suitable message, after all student names have been displayed. |                             **[6]** | |

**END OF QUESTION PAPER**

# Mark scheme

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Question** | | | **Answer/Indicative content** | **Marks** | **Guidance** |
| 1 | a |  | 1 mark for each letter     |  |  | | --- | --- | | Decomposition | D | | Abstraction | B | | Input Sanitisation | A | | Casting | F | | 4  AO1 1a(4) | Accept answers that write the definition instead of the letter. |
|  | b | i | * timer = 7.3 | 1  AO3 2b(1) | Ignore dim / define / etc and data types Do not allow use of string delimiters or other unsuitable data types. Allow other suitable assignment symbols (e.g. := ) Do not allow == for assignment. Do not penalise case. Spelling must be accurate |
|  |  | ii | * Real / Float | 1  AO2 1b(1) | Allow decimal, single, double or equivalent |
|  |  |  | **Total** | **6** |  |
| 2 | a |  | |  |  |  | | --- | --- | --- | | Line | Program code | output | | 08 | print (score) | **18** | | 09 | print ("name") | **name** | | 10 | print (newscore(score,2)) | **37** | | 11 | print (score) | **18** | | 4  AO2 1b(4) | **Examiner’s Comments** Question (a) gave candidates pseudocode (including a function) and were asked to state the output values from various lines. Most candidates were able to do this for more simple output. Fewer candidates completed the function call and return values accurately.  Many candidates struggled to differentiate between name (as an identifier for a variable) and “name” (as a string).     |  |  | | --- | --- | |  | **Misconception** |   print(“name”) will output the literal string “name” to the screen”.  print(name) will print the contents of the variable with identifier name. |
|  | b |  | 1 mark per bullet to max 2   * **Easier/quicker** for humans to **write** * **Easier/quicker** to **read / understand / remember** * **Easier/quicker** to **maintain / debug / spot errors** * …because code is closer to English / uses English words * Less code to write * …because one HLL instruction represents many assembly instructions * Portable (between processors) / will work with different types of compute | 2  AO1 1b(2) | Accept “human language” as English for BP4  “Easier to use” is too vague.  **Examiner’s Comments** Question (b) was answered extremely well. Many candidates were able to articulate the advantages of high-level languages over assembly language. Even more pleasing were the many candidates able to discuss issues such as portability of code and that a high-level instruction translates into many assembly instructions. |
|  | c |  | 1 mark per bullet to max 2   * Each character (in character set) has a **unique** (binary) number/value * Each character in the **string** is assigned its associated number/value * The (binary) value of each character is stored/combined (in order) * … by example e.g. The binary value for D, then for r, then for u * Uses ASCII/Extended ASCII/Unicode | 2  AO2 1a(2) |  |
|  |  |  | **Total** | **8** |  |
| 3 | a | i | 1 mark per bullet to max 2 e.g.   * Check the program meets the **user requirements** * Check the program works (as intended) / detect logic / syntax errors * Check the program does not crash (under invalid entry) / check error messages are suitable * …allow these errors to be fixed * …make sure there are no problems when released * Any suitable example **related to the vending machine** e.g. gives correct change | 2  AO1 1b(2) | Allow two any suitable examples for two marks  BOD “find errors”, “find bugs” for BP2  “fix errors” by itself is one mark (BP4) |
|  |  | ii | 1 mark per bullet to max 2   * Iterative is during development / repeatedly testing after/while making changes * Final is when the development is (almost) complete / done after iterative testing | 2  AO1 1b(2) | Do not accept just “repeatedly testing” for iterative  BOD “iterative testing tests modules/sections” |
|  |  | iii | |  |  |  | | --- | --- | --- | | Code entered | Money inserted | Expected result | |  |  |  | | **C2** |  |  | |  | **£0.49** (or any value less that £0.50) |  | |  |  | **Invalid Selection** (or any suitable error message) | | 3  AO3 2b(3) | For £0.49 accept any value <£0.50. Must be a specific value, not a description.  Accept any suitable error message for invalid selection |
|  | b | i | * money * price | 1  AO1 1b(1) | Must be an identifier, not description. Ignore case. |
|  |  | ii | * one | 1  AO2 1b(1) | **Examiner’s Comments** Question (b)(i) was answered very well. However, (b)(ii) was only answered correctly by a small proportion of candidates. These were also the candidates who generally went on to achieve high marks on this paper.  The question asked candidates to state how many parameters are passed into the function from the line giveChange(money – price). Two variables are inside the brackets. Candidates did not recognise this as a calculation.  This calculation would be completed before the function call. Only the result is passed into the function as the parameter. This means that the correct answer is one. Most candidates gave the (incorrect) answer of two.     |  |  | | --- | --- | |  | **Misconception** |   Where a sub program (function or procedure) has multiple parameters passed into it, there will be separated by commas – for example testfunction(x,y). A subprogram with a calculation as a parameter will pass the result of this calculation into the sub program. |
|  | c |  | 1 mark per bullet   * Checking if money>=price… * …decision (diamond shape) used * …venditem() and giveChange(money-price) if True/Yes * …output an error if False / No * Terminator used to start and end the program and all paths terminated | 5  AO3 2b(5) | Reasonable attempt at BP1 needed for credit BP2, 3 and 4 Ignore other additional code.  BP3 and BP4 must follow on from True/False / Yes/No decision to be credited.  Subroutines names and parameters must be correct. Ignore missing brackets on venditem.    **Examiner’s Comments** Question (c) asked candidates to draw flowchart and this was done particularly well. Many candidates changed the given pseudocode algorithm into a well-defined process that covered the same steps as the pseudocode.  A typical answer for this question would be good to use for centres when teaching about using flowcharts. Candidates who use flowcharts for algorithm answers are often not specific enough with the steps required. This gives answers that are generic or high-level. The structure of this question is a good example of the level of detail required. |
|  | d | i | 1 mark per bullet to max 2   * Indentation / whitespace * Appropriately named variables / identifiers * Modularisation / use of subroutines | 2  AO2 1b(2) |  |
|  |  | ii | * Comments * Use of constants | 1  AO2 1b(1) |  |
|  | e |  | * SELECT ItemCode / \* * FROM ITEMS * WHERE * …Stock < 10 | 4  AO3 2b(4) | Accept other fields shown in addition to ItemCode  Accept Stock <=9 / etc.  Ignore case. Spelling of fields and table must be correct.  If WHERE missing, Stock < 10 must be after FROM clause.  **Examiner’s Comments** Question (e) covered SQL. It was clear that many candidates were not sure what was being asked for in this question. Many incorrectly attempted to use keywords such as IF and THEN. The use of SQL in this specification is limited to a small number of keywords, as listed in the specification.     |  |  | | --- | --- | |  | **OCR support** |   Page 43 of the specification lists the SQL keywords that should be covered. These are limited to SELECT, FROM, WHERE, LIKE, AND, OR and wildcards.  The current specification is available here : https:/www.ocr.org.uk/images/225975-specification-accredited-gcse-computer-science-j276.pdf |
|  | f |  | 1 mark per bullet   * Input from user * Check IF input value is “**on**”… * … if so, assign **1** to statevalue * Correct assignment of **2 for “off**” and **3 for “suspended”** with correct state and IF * Correct logical check (else) to output “invalid state” if no state set | 5  AO3 2b(5) | Accept alternative error messages. Variable names must not include obvious spaces.  BP3 dependent on BP2. BP2 and BP4 must be a logical comparison using IF and not just the CASE statement. NE to simply replace CASE with IF.  Penalise each error once then apply FT.  e.g. newstate = input("Enter the new state : ") if newstate == "on" then    statevalue = 1 elseif newstate = "off" then    statevalue = 2 elseif newstate = "suspended"    statevalue = 3 else    print("Invalid state") endif  **Examiner’s Comments** Question (f) asked candidate to rewrite an algorithm that used switch/case to perform the same actions but using IF statements. It was clear that many candidates did not understand the use of switch/case. Some languages (such as Python) do not include support for these. It is important that these constructs are still taught.     |  |  | | --- | --- | |  | **OCR support** |   Appendix 5f of the GCSE Computer Science specification covers the pseudocode guide for this examination. Candidates do not need to use this in their responses but they should be aware of this as questions will be presented using this format.  The current specification is available here :  https:/www.ocr.org.uk/images/225975-specification-accredited-gcse-computer-science-j276.pdf |
|  |  |  | **Total** | **26** |  |
| 4 | a |  | * E 3 | 2  AO1 1b(2) | 1 mark per digit (mark right to left) Max 1 if any additional leading values |
|  | b |  | * 0110 1001 must be 8 bits | 2  AO1 1b(2) | 1 mark per nibble (mark right to left). Max 1 if any additional leading values |
|  | c |  | 1 mark per bullet to max 2   * Easier/quicker to communicate / enter / write / read / remember * Less chance of input errors / easier to spot errors * They are smaller / shorter * Easy to convert between binary and Hexadecimal | 2  AO1 1b(2) | Mark response as a whole.  Do not accept answers simply describing what hexadecimal is.  “easier to understand” or “easier to use” on its own is NE  BP3 (smaller) must refer to size when written down, NOT size when stored which is unaffected) |
|  | d | i | * 3 | 1  AO1 1b(1) | CAO |
|  |  | ii | * 1 | 1  AO1 1b(1) | CAO |
|  | e |  | * 00001111 | 1  AO1 1b(1) | Ignore missing or additional leading zeros |
|  | f | i | 1 mark per bullet point   * B AND C * OR gate with two inputs, one of which is A * …correct connection of these two gates with no additional gates / connections | 3  AO1 1b(3) | Shape must be accurate |
|  |  | ii | 1 mark per bullet point   * Correct completion of A and B inputs as **1 1** * 0 output for 01 input * 0 output for 10 input * 0 output for 11 input | 4  AO1 1b(1)  AO2 1b(3) | CAO     |  |  |  | | --- | --- | --- | | **A** | **B** | **P** | | **0** | **0** | **1** | | **0** | **1** | **0** | | **1** | **0** | **0** | | **1** | **1** | **0** | |
|  |  |  | **Total** | **16** |  |
| 5 | a |  | * Number of pixels (in an image) * Height and width (of an image) | 1  AO2 1b(1) | Accept pixels per inch / mm / unit area (density) |
|  | b |  | * 90 (pixels in an image) / 15 x 6 (pixels in image) * Multiply pixels x bits per pixel * …2 bits required per pixel (because 3 colours) * 180 bits overall answer | 4  AO1 1b(2)  AO1 1b(2) | Must clearly show multiplication for 3rd BP |
|  | c |  | * Reduce number of pixels / resolution * Reduce number of colours * Use lossy compression * Use lossless compression | 2  AO2 1a(2) | Accept descriptive answers linked to given logo (e.g "change to black and white only") “Make image smaller” is NE  Allow compression by itself for one answer. |
|  | d | i | * Data about data / the image/file / properties of the file | 1  AO1 1b(2) | Do not accept examples without a definition. |
|  |  | ii | e.g.   * height * width * colour depth * resolution * geolocation * date/time created/last edited / timestamp * file type * author details | 1  AO1 1a(2) | Accept any sensible data that could be stored alongside an image.  Do not accept filename |
|  |  |  | **Total** | **9** |  |
| 6 | a |  | 1 mark per bullet to max 6   * Access “Rob” / studentnames[0]… * …does **not** equal “Anna” / not desired item / move on * Access “Anna” / studentnames[1] * …does equal “Anna” / stop / item found | 4  AO2 1b(4) | Answer must refer to this array, not a generic description of linear search. “Access first item” is NE for BP1 or BP3. Must refer to this scenario.  Max 1 for “Compare ‘Anna’ to each item in list” if nothing else credited. |
|  | b |  | * Anna inserted before Rob as first two elements… * …Huw correctly inserted into sorted list… * …Emma correctly inserted into sorted list … * …Patrice correctly inserted into sorted list … * …Iqbal correctly inserted into sorted list and no further changes made. | 5  AO2 1b(5) | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Rob | Anna | Huw | Emma | Patrice | Iqbal | | **Anna** | Rob | Huw | Emma | Patrice | Iqbal | | Anna | **Huw** | Rob | Emma | Patrice | Iqbal | | Anna | **Emma** | Huw | Rob | Patrice | Iqbal | | Anna | Emma | Huw | **Patrice** | Rob | Iqbal | | Anna | Emma | Huw | **Iqbal** | Patrice | Rob |   Sorted list highlighted |
|  | c |  | * Use of iteration (any use) … * …loops for each item in array / loops 6 times * …to print out each item in studentnames * …input attendance * Add up/calculate students present and absent * …Outputs present and absent (in suitable message) | 6  AO3 2b(6) | BP 2 and 3 may be met together with suitable input statement. Both dependent on attempt at iteration.  BP5 not dependent on correct previous parts.  BP6 needs reasonable attempt at totalling present and absent figures.  Ignore non-initialisation of counter variables.  Flowcharts are acceptable but must show how to solve the problem, not simply repeat the question.  Example algorithm present=0 absent=0 for i = 0 to (studentnames.length) -1    print(studentnames[i])    attendance=input("absent or present?")    if attendance=="present" then     present=present+1    else     absent=absent+1    endif next i print ("Present students: " + present) print ("Absent students: " + absent)  **Examiner’s Comments** Question (c) asked candidates to write an algorithm to:  1. Call an attendance register from a given array. 2. Count and output how many students were present and absent.  It was pleasing to see candidates decompose the problem and tackle each part to build a full solution.  The bullet points given in the question served as a scaffold.  Where one section was incorrect, other marks could be given.  A number of candidates attempted to use a FOR…IN loop to iterate over the given student array.  FOR…IN is not listed in the specification but is an entirely suitable way to approach the problem. Where this was logically sound, full marks were available.  Other candidates used FOR…NEXT loops or WHILE loops in conjunction with the length of the array. Again, marks were given where the solution was logically sound.  Marks were more limited for those candidates who did not attempt any sort of loop, but some were still able to be given where appropriate.     |  |  | | --- | --- | |  |  |   Centres should be encouraged to connect together content within the specification, so that rather than teaching arrays in isolation, they could perhaps be combined with iteration to be able to count up or total values in an array, or apply this to how searching and sorting algorithms work. Where this is done on a regular basis, questions such as 6(c) perhaps become less daunting for candidates. |
|  |  |  | **Total** | **15** |  |