GCSE 9-1

Computer Science

Mark Scheme for Paper 2.2

Component 2 - Computational Thinking, Algorithms and Programming

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Question** | | | **Answer** | **Mark** | **Guidance** |
| 1 | a |  | 1 mark per bullet max 3   * An image is made of pixels * Each pixel can be one colour * Each colour has a unique/corresponding binary number * Each pixel is given the binary number of its colour * The binary numbers are stored in order in the file * E.g. white = 0, black = 1, first line would be 10001 | 3 |  |
| 1 | b |  | * Number of bits used to store (the colour of) each pixel | 1 |  |
| 1 | c |  | * Concentration of pixels | 1 |  |
| 1 | d |  | * 2 * bits | 2 | One mark for correct number, one mark for correct unit |
| 1 | e |  | May use answer carried forward from part 1d.   * One mark for calculating correct resolution: 36 pixels * One mark for resolution x colour depth: 72 bits (or 36 x their colour depth) * One mark for converting 72 bits (or their answer) to bytes: 9 bytes | 3 |  |
| 2 | a |  | * SELECT Name * FROM Lessons * WHERE LessonType = “Group”   One mark per correct line. | 3 |  |
| 2 | b |  | * Name: String * NumberOfLessons: Integer * Paid: Boolean   One mark per correct line. | 3 |  |
| 2 | c |  | * 115 | 1 |  |
| 2 | d |  | * 35 | 1 |  |
| 2 | e |  | 1 mark per bullet to max 8  Sum of all numbers in the array:   * Creation of variable to store total * Creating a loop that loops through items in the list * Adding each item in the list to the total variable   Highest number in array:  EITHER   * Making a variable to store highest number, setting it to a low number * Looping through each item in array * Checking IF item is bigger than highest number stored, if it is, replace highest number with current number in list   OR   * Sort list * Get last item in list (or first item if sorted largest to smallest) * Store this item as highest number   Outputs   * Outputting total and highest number | 7 | **Example algorithms:**  **For finding total:**  total = 0  For x = 0 to 6:  total = total + payments[x]  print(total)  **For finding largest number:**  **Example 1:**  highest = 0  for x = 0 to 6:  if payments[x] > highest:  highest = payments[x]  print(highest)  **Example 2:**  payments.sort()  highest = payments[6]  print(highest) |
| 3 | a |  | * 32 + 16 + 8 + 4 + 2 + 1 * 0011 1111 | 2 |  |
| 3 | b |  | 1 mark per hex digit   * First nibble = 2 + 1 = 3 * Second nibble = 8 + 4 + 1 = 13 = D * Answer = 3 D | 2 |  |
| 3 | c |  | * Answer 0 1 1 1 0 0 1 1 * One mark per nibble | 2 |  |
| 3 | d |  | 1 mark per bullet to max 2   * There is an extra bit * Answer cannot fit into 8 bits * Result is greater than 255/11111111 | 2 |  |
| 4 | a |  | 1 mark per bullet   * Name of first symbol: NOT * Result of first symbol: 1 * Name of first symbol: AND * Result of first symbol: 0 | 4 |  |
| 4 | b |  | 1 mark for correct name of each   * Left symbol: NOT * Right symbol: OR   1 mark per correct row in table. Do not have to be in order. Do not give marks for repeated rows.   |  |  |  | | --- | --- | --- | | **A** | **B** | **OUT** | | 0 | 0 | 1 | | 0 | 1 | 1 | | 1 | 0 | 0 | | 1 | 1 | 1 | | 6 |  |
| 5 | a | i | For first mark:   * Logic error - Program does something other than what the programmer intended   1 mark per bullet to max 2   * totalProfit variable is inside the loop * totalProfit is reset to 0 every time loop repeats * totalProfit at the end of the loop will not return the profit for the week, only the last day | 3 |  |
| 5 | a | ii | * Line number 7 * Line should have Primt changed to print. | 2 |  |
| 5 | b |  | 2 marks for describing a variable, 1 mark for giving an example   * A name which represents a value in a program * ...points to a memory location * ...value can be changed while the program is running. * Examples:   + totalProfit   + dayEarnings   + dayCosts | 3 |  |
| 5 | c |  | 1 mark for each correct row.   |  |  | | --- | --- | | **Sequence** | ✔ | | **Selection** |  | | **Iteration** | ✔ | |  |  |
| 5 | d | i | * Any number |  |  |
| 5 | d | ii | * A letter/character that isn’t a number |  |  |
| 5 | e |  | 1 mark for naming method, 1 mark for explanation:   * Comments * ...can be used to explain how code works * ...describe purpose of a part of code * ...explain purpose of program * ...can say who wrote the program and when * Use naming conventions/meaningful variable names * ...easier to understand purpose of variable/what variable stores * Use subprograms/subroutines * ...split program up into specific tasks * ...easier to locate and amend one task | 2 |  |
| 6 | a |  | 1 mark per bullet to max 4   * List split into individual elements (may be done over several steps or just as a starting point) * Merge individual elements into sorted lists of size 2 * Merge lists of size 2 into sorted lists of size 4 * Merge lists of size 4 into final sorted list. | 4 | Can describe how the merge sort works or show with a diagram the list at each stage.  Answers showing/describing other sorting algorithms are awarded 0 marks. |
| 6 | b |  | * Uses less memory (as only one list needs to be stored) | 1 | Can name any other valid benefit |
|  |  |  | * More efficient | 1 | Can name any other valid benefit |
| 7 | a |  | 1 mark per bullet, max 2   * Input sanitisation * Input validation * Authentication * Anticipating misuse | 2 |  |
| 7 | b |  | 1 mark for feature, 1 mark for explanation   * Editor * ...allows you to write and edit code * ...colour coding allows you to identify certain elements easier * ...auto-correct corrects syntax errors making programming more efficient * ...auto-suggestion reminds you of how to write certain elements of program * ...automatic line numbering makes it easier to identify specific lines * ...auto-indent makes programming more efficient * Error Diagnostics/debugging * ...tells you where an error is in a program so it can be fixed * ...tells you the type of error to help you fix it * ...suggests solutions * Run-Time Environment * ...enables the program to be run * ...tests the program * Translator/compiler/interpreter * ...converts high level code into machine code/low level code/binary * ....enables the code to be executed/run | 4 |  |
| 7 | c |  | * Asking the user for their first name and surname and storing them in appropriate variables * Using appropriate string manipulation to get the first 2 letters of first name * Using appropriate string manipulation to get the first 3 letters of last name * Concatenating the parts of the first name and last name together to create a username * Outputting the username. * Asking the user to enter a password and storing it * Finding out the length of the password * Using an “if” statement to check if the length of the password is <10 and outputting “too short” if it is * Using an “else” (or possibly another “if”) to output “account created” if password is suitable | 9 | Award marks for using a “while” loop for checking the password.  Example answer:  firstName = input("What is your first name?")  surname = input("What is your surname?")  part1 = firstname.subString(0,2)  part2 = surname.substring(0,3)  username = firstname.subString(0,2) + surname.substring(0,3)  print("Username is: " + username)  password = input("Enter a password: ")  if password.length < 10 then  print("too short")  else  print("account created")  endif |

**School of Coding**

8 Newton Court

Pendeford Business Park

Pendeford

Wolverhampton

WV9 5HB

**Contact Details:**

Telephone: 01902 509 209

Email: info@schoolofcoding.co.uk

[www.schoolofcodinguk.com](http://www.schoolofcodinguk.com)

