Prepare for your exam

Paper 1 – Principles of computer science

Paper 2 – Application of computational thinking

Paper 2 – Application of Computational Thinking

This paper is worth 50 per cent of your GCSE Computer Science qualification. It focuses on Topic 1 and Topic 6 of the subject content of the specification and assesses your computational thinking and problem-solving skills. It is 2 hours in length.

You must have:

- a computer workstation with appropriate programming language code editing software and tools, including an IDE that you are familiar with which shows line numbers
- a 'STUDENT CODING' folder containing code and provided data files
- printed and electronic copies of the Programming Language Subset (PLS) document.

Exam tip

- Save copies of the original files. Frequently save your modification through the exam. That way, if you mess up your code badly, you can go back to the last good copy.
- Each question has a suggested time. You may need slightly less or slightly more, so make sure you balance out your time.
- If you get stuck on a question, you should move on to the next. You can always come back if you have time at the end.
- If you get stuck on part of a question, think how you could get around it and still earn marks. For example, if you have to write validation for input before processing it, but you can't get the validation to work, skip it and write the code for processing it.
- If a part of your code doesn't work or run, then comment it out. This way, the remainder will actually run.

Understanding the questions

More information may be given at the start of each question. Read this carefully. In some cases, you won't be able to answer the question without referring to it.

All the questions are compulsory.

Here are the command words you might encounter in this paper.

Amend. Making changes. For example, amending an algorithm so that it performs differently or in order to correct an error. It also includes adding to code, deleting from code, or arranging lines of existing code.

Write. For example, creating or manipulating program code.

Open. While not a command word, this means to open the named file in your editor or IDE.

Save. While not a command word, this means to save your amended files onto your secure exam profile user area.

Question 01

Suggested time: 10 minutes

1 A program adds two numbers, entered by the user. The program also displays the user's name to the screen.

Open file **Q01.py**

Amend the code to add lines to:

- create three integer variables
- create one string variable
- ask the user to enter two numbers
- ask the user to enter their name
- add the two numbers and save it into another variable
- display the sum and user's name, on the same line.

Do **not** add any additional functionality.

Save your amended code file as **Q01FINISHED.py**

(Total for Question 1 = 7 marks)

Provided

```
# ------------------------------------------------------------------------------
# Global variables
# ------------------------------------------------------------------------------
# =====> Create three global variables named 'num1', 'num2', and 'num3'.
#        Initialise them to sensible values.

# =====> Create a global variable named 'name' and initialise it to a sensible
#        value

# ------------------------------------------------------------------------------
# Main program
# ------------------------------------------------------------------------------
# =====> Ask the user to enter an integer value and save it into the variable
#        'num1'

# =====> Ask the user to enter an integer value and save it into the variable
#        'num2'

# =====> Ask the user to enter a string value and save it into the variable
#        'name'

# =====> Add the two numbers and assign it to 'num3'

# =====> Display 'num3' and 'name' to the screen on the same line
```

Mark scheme

- Declaration and initialisation of num1, num2, num3 to sensible integer value (1)
- Declaration and initialisation of name to a sensible string value (1)
- 2 x Using input with a prompt to get numbers from user (1)
- 2 x Converting string input to integer before storage (1)
- Using input with prompt to get name from user (1)
- Adding two numbers and assigning it to the third variable (1)
- Using print function with comma to show items on one line (1)

Response

```
# ---------------------------------------------------------------------------
# Global variables
# ---------------------------------------------------------------------------
# =====> Create three global variables named 'num1', 'num2',and 'num3'.
#        Initialise them to sensible values.
num1 = 1
num2 = 2
num3 = 3

# =====> Create a global variable named 'name' and initialise it to a sensible
#        value
name = "name"

# ---------------------------------------------------------------------------
# Main program
# ---------------------------------------------------------------------------
# =====> Ask the user to enter an integer value and save it into the variable
#        'num1'
num1 =int(input("Enter number: "))

# =====> Ask the user to enter an integer value and save it into the variable
#        'num2'
num2 = int(input("Enter number: ")

# =====> Ask the user to enter a string value and save it into the variable
#        'name'
name = input("Enter your name: ")
# =====> Add the two numbers and assign it to 'num3'
num3 = num1+num2

# =====> Display 'num3' and 'name' to the screen on the same line
print (total)(name, num3)
```

Verdict

Declaration and initialisation of num1, num2, and num3 on lines 6 to 7. Values of 0, might have been better, but as they're integer values, as required by the program, and are going to be immediately overwritten, they earn (1) mark.

Declaration of variable 'name' and initialised to a string, as required by the program, earns (1) mark.

Using input() with an appropriate prompt, twice, earns (1) mark.

Converting string inputs to integers, twice, earns (1) mark.

Using input() with appropriate prompt to receive string earns (1) mark.

Adding the two inputs and assigning it to the third variable earns (1) mark.

Using print() to output name and number on same line earns (1) mark. This example has been done with a comma, but would be awarded if using concatenation with conversion of num3 to string.

7 out of 7 marks for this response.

Exam tip

This question just requires you to follow instructions and write the lines of code required under the comments.

There may be more than one way to write the code lines that answer the question.

Fix all your syntax errors. Be sure to run the code to make sure it actually functions properly.

Question 02

Suggested time: 20 minutes

2 A programmer has started to write a program, but it does not work correctly. The console session output of the corrected program is shown here.

```
C:\Code\PycharmEnv\Scripts\python.exe
4
6
Process finished with exit code 0
```

Open file **Q02.py**

Amend the code to:

- fix the syntax error on the original line 1
- fix the syntax error on the original line 7

- fix the syntax error on the original line 10
- change the identifier 'a' to a more meaningful name
- change the identifiers 'count' and 'count2' to more meaningful names
- fix the runtime error caused by the original line 7
- fix the logic error which causes the incorrect output
- add a comment to explain the original line 6
- add at least one use of white space to aid readability
- ensure that the program produces the required output.

Do **not** add any additional functionality.

Save your amended code file as **Q02FINISHED.py**

(Total for Question 2 = 10 marks)

Provided

```
a = ["M", "M", "M", "M", "F", "F", "M", "F", "M", "M"]
length = 0
count = 0
count2 = 0
index = 0
length = len(a)
while (index <= lenght)
    if (a[index] == "F")
        count = count + 1
    ese:
        count2 = count2 + 1
    index = index + 1
print (index)
print (count2)
```

Mark scheme

- Syntax – add double quote to gender list item (1)
- Syntax – add ':' to end of WHILE statement (1)
- Syntax – change 'ese:' to 'else:' (1)
- Meaningful identifier for gender list (1)
- Meaningful identifier for count and count2 (1)
- Runtime error – change '<=' to '<' to avoid IndexError (1)
- Logic error – change 'print(index)' to 'print(count)' (1)
  - Readability – add some white space to separate variables from main program (1)
  - Readability – add any meaningful comment to explain logic (1)
  - Functionality – working solution (1)

Response

```
letters = ["M", "M", "F", "M", "F", "F", "M", "F", "M", "M"]
length = 0
countF = 0
countM = 0
index = 0
length = len(letters)     #get the length of the list so that we can use in
                          #the loop

while (index <= lenght)
    if (letter[index] == "F")
        countF = countF + 1
    else:
        countM = countM + 1
    index = index + 1


print (countF)
print (countM)
```

Verdict

Missing double quote added to gender array, earns (1) mark.

Adding ':' to end of while loop, earns (1) mark.

Correct spelling of 'else', earns (1) mark.

Naming of list as 'letters' is acceptable as problem doesn't say they're gender, earns (1) mark.

Both count variables are tied to letters in the list to make them more meaningful, earns (1) mark.

Correction of runtime error (<= to <) on line 8, earns (1) mark.

Correction of logic error, by printing correct count, earns (1) mark.

White space added to separate declarations and assignments from code, to separate output statements at the bottom, and block out loop. This is an excellent example of how to use white space to make code more readable, earns (1) mark.

Good comment on line 6, connecting the len() function to the loop, earns (1) mark.

The code runs in the interpreter and produces the required result, earns (1) mark.

10 out of 10 marks for this response.