Paper Reference 1CP2/02

Pearson Edexcel
Level 1/Level 2 GCSE (9-1)

COMPUTER SCIENCE
PAPER 2: APPLICATION OF COMPUTATIONAL THINKING

Time: 2 hours

Q71103A

Pearson

YOU MUST HAVE:

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

YOU WILL BE GIVEN

- **A separate Data Book.**

You do not need any other materials.

(continued on the next page)

INSTRUCTIONS

- Answer ALL questions on your computer.
- Save the new or amended code using the file name provided and place it in the 'COMPLETED CODING' folder.
- You must NOT use the internet at any time during the examination.

INFORMATION

- The 'STUDENT CODING' folder in your user area includes all the code and data files you need.
- The total mark for this paper is 75.
- The marks for EACH question are shown in brackets.

ADVICE

- Read each question carefully before you start to answer it.
- Save your work regularly.
- Check your answers and work if you have time at the end.

ANSWER ALL QUESTIONS.

SUGGESTED TIME: 15 MINUTES

1. **A program is required to convert numbers entered by the user to their alphabetic equivalent.**
   Only numbers from 5 to 30 are valid.
   Adding 60 to the number and then applying the function chr() generates the equivalent ASCII code for an uppercase letter.

   The table below shows accurate test data for a functional program.

INPUT	OUTPUT
4	Invalid input
5	5 is equal to A
22	22 is equal to R
30	30 is equal to Z
31	Invalid input

(continued on the next page)

Question 1 continued

Open file Q01.py

Amend the code to:

- **create an integer variable named num and set it to 0**
- **take the input from the user and convert it to an integer**
- **check that the inputted number is between 5 and 30**
- **add 60 to the variable num and assign the result to the variable decimalCode**
- **join strings together with concatenation**
- **display an error message.**

Do NOT add any additional functionality.

Save your amended code file as

Q01FINISHED.py

(Total for Question 1 = 10 marks)

SUGGESTED TIME: 15 MINUTES

2. A program uses turtle graphics to draw a simple image.

Look at the Image for Question 2 in the separate Data Book.

It shows the image that must be produced.

Both the circle and the outside square are centred on the horizontal and vertical grid lines.

The outside square is 400 × 400

The circle is 200 across. The circle outline is coloured gold. All other lines are black.

The program has errors and does not work correctly.

(continued on the next page)

Question 2 continued

Open file Q02.py

Amend the code to:

- **add a comment to identify the data type of the argument to the turtle.mode () subprogram call on original line 19**

  turtle.mode ("standard")
- **fix the NameError on original line 23**

  screen.setup (WIDTH, HIGHT)
- **fix the AttributeError on original line 28**

  theTurtle = turtle.turtle ()
- **fix the TypeError on original line 36**

  theTurtle.pendown (200)
- **fix the logic error that causes the vertical axis to be too far right on original line 42**

  theTurtle.setpos (100, 200)
- **fix the logic error that causes the vertical axis to be drawn too short on original line 48**

  theTurtle.forward (100)

(continued on the next page)

Question 2 continued

- **fix the logic error that causes the outside square to tilt left of the vertical axis on original line 56**
  theTurtle.setheading (95)
- **add a line to set the size of the pen to the constant BIG on original line 68**
- **add a line to set the colour of the pen to gold on original line 71**
- **add a line to hide the turtle on original line 78.**

Do NOT change the functionality of the given lines of code.

Do NOT add any additional functionality.

Save your amended code file as
Q02FINISHED.py

(Total for Question 2 = 10 marks)

SUGGESTED TIME: 20 MINUTES

3. A circle is cut out of a square of card.

A program is needed to calculate the area of the excess card.

Look at the Image for Question 3 in the separate Data Book. The image shows the excess card in light grey.

The required formulae are:

$$\text{area of a square} = s^2$$

$$\text{diameter of a circle} = 2r$$

$$\text{area of a circle} = \pi r^2$$

- ***s*** **is the length of a single side of the square**
- ***r*** **is the radius of the circle, the distance from the centre to the edge**
- **π is the constant Pi.**

(continued on the next page)

Question 3 continued

The table below shows accurate test results for two sets of inputs.

INPUT		OUTPUT
LENGTH OF SIDE	RADIUS OF CIRCLE	
12	8	Invalid input
10	4	49.73451754256331

The program has these requirements:

- **the length of the square's side is an integer**
- **the radius of the circle is an integer**
- **the user inputs both the length of a side and the radius (no validation required)**
- **the diameter of the circle must be the same size or smaller than the side of the square**
- **the variables supplied in the code file must be used.**

(continued on the next page)

Question 3 continued

Open file **Q03.py**

Amend the code to meet the requirements.

Do NOT change the functionality of the given lines of code.

Do NOT add any additional functionality.

Save your amended code file as

Q03FINISHED.py

(Total for Question 3 = 10 marks)

SUGGESTED TIME: 20 MINUTES

4. A program converts a binary pattern to a positive denary integer. The user enters the binary pattern. The program loops continually until the user inputs an empty pattern to stop the program.

   A multiplier doubles for each position from right to left, to generate the place values for the binary pattern, i.e. 128, 64, 32, 16, 8, 4, 2, 1. When a 1 appears in the pattern, the place value is added to the running total. When a 0 appears in the pattern, nothing is added to the running total.

   The table below shows accurate test results for three inputs.

INPUT	OUTPUT
1100	12
10101010	170
<empty>	Exits program

(continued on the next page)

Question 4 continued

The lines of code in the program are mixed up.

The indentation has been done for you.

Open file Q04.py

Amend the code to make the program work and produce the correct output. You will need to rearrange the lines.

Do NOT change the functionality of the given lines of code.

Do NOT add any additional functionality.

Save your amended code as Q04FINISHED.py

(Total for Question 4 = 15 marks)

SUGGESTED TIME: 20 MINUTES

5. **A program uses a one – dimensional data structure to record the number of kilograms of coffee used each day by the local cafe. The data needs to be saved to a file.**

   Each line of the file represents one week's worth of coffee consumption.

Open file Q05.py

Write a program to meet these requirements:

- **create a comma – separated value text file named Q05_OUTPUT.TXT**
- **write lines to the file, each holding seven weights**
- **process all the weights in the data structure.**

(continued on the next page)

Question 5 continued

Look at the Output file for Question 5 in the separate Data Book.

It shows the expected output file.

Do NOT add any additional functionality.

Use comments, white space, indentation and layout to make the program easier to read and understand.

Save your amended code as Q05FINISHED.py

(Total for Question 5 = 15 marks)

SUGGESTED TIME: 30 MINUTES

6. In a board game, users build words from letter tiles.

Each letter tile has a points value.

There are 107 valid two–letter words.

A program is required to check that each two–letter word placed on the board is valid.

Each valid two–letter word with its points value is stored as a record in the two–dimensional list, **wordTable.**

The records are stored in alphabetical order.

Open file **Q06.py**

Write a program to meet these requirements:

INPUTS

- prompt for and accept a two–letter word from the user, AA to ZZ, inclusive
  - accept uppercase and lowercase input
  - no other validation is required.

(continued on the next page)

Question 6 continued

PROCESS

- create a linear search to locate the word in wordTable
  - stop the search when:
    - the word is located
    - the expected location of the user's word is passed
    - the end of the list is reached after all words have been checked
  - ensure the search works for any length of wordTable

OUTPUTS

- when the user's word is located, inform the user of the word and the number of points it scores
- when the expected location of the user's word is passed, suggest the next word and the number of points it scores
- when the end of the list is reached after all words have been checked, suggest the last word in the list and the number of points it scores.

(continued on the next page)

Question 6 continued

Use comments, white space and layout to make the program easier to read and understand.

Do NOT add any additional functionality.

Save your amended code as Q06FINISHED.py

(Total for Question 6 = 15 marks)

TOTAL FOR PAPER = 75 MARKS

END OF PAPER