2025L219A2EL

Coimisiún na Scrúduithe Stáit
State Examinations Commission

Leaving Certificate Examination 2025

Computer Science

Section C

Higher Level

Wednesday 21 May Morning 11:30 – 12:30

80 marks

Do not hand this up.

This document will not be returned to the State Examinations Commission.

Instructions

There is **one** section in this paper.

Section C	Programming	**One** question Answer **all** question parts	80 marks

Answer **all** parts of the question on your digital device.

Calculators may be used during this section of the examination.

The *Formulae and Tables* booklet cannot be used for this section of the examination.

The superintendent will give you a copy of the *Python Reference Guide*.

Ensure that you save your work regularly.

Save your files using the naming structure described at the beginning of each question part.

If you are unable to get some code to work correctly, you can comment out the code so that you can proceed. The code that has been commented out will be reviewed by the examiner.

Rough work pages are provided at the end of this booklet. Please note that this booklet is not to be handed up and will **not** be reviewed by an examiner.

At the end of the examination it is your responsibility to ensure that you have saved your files onto your external media.

You will be provided with a brown envelope for your external media. Write your examination number on this envelope and place your external media into it before sealing. Place this envelope in the pouch at the front of the red envelope that contains your examination booklet from Section A and B.

There is no examination material on this page

Section C	Programming	80 marks

Answer **all** question parts.

Question 16

(a) Open the program called **Question16_A.py** from your device. The source code is shown below and described on the next page.

Before making any changes, you should save your working copy of the file using the format **ExaminationNumberQuestion16_A.py**. For example, you would save the file as **123456Question16_A.py** if your Examination Number was 123456.

Enter your Examination Number in the space provided on **line 2** in your Python file**.**

```
1  # Question 16 (a)
2  # Examination Number:
3
4  def get_grade(result):
5      grade = "Unsuccessful"
6
7      if result >= 80:
8          grade = "Distinction"
9      elif result >= 65:
10         grade = "Upper Merit"
11
12     return grade
13
14 # Calculate and display the mean of a list of results
15 results = [39,32,62,88,51,62,64,81,77] # Initialise the list
16 N = len(results) # Initialise N to the number of results
17 total = 0 # Initialise the running total to 0
18
19 # Loop N times
20 for i in range(N):
21     total = total + results[i] # Running total
22
23 # Divide by the total number of results to give the mean
24 arithmetic_mean = total/9
25
26 # Display the answer
27 print("The mean percentage mark is", arithmetic_mean)
```

Line 15 of the program initialises a list called `results` with nine values. Each value represents a percentage mark obtained by an individual student in nine class tests all in the same subject. The final percentage for the subject is calculated by averaging the nine results.

The `for` loop adds up the values in `results` and stores the answer in the variable called `total`. Line 24 of the program calculates the mean (average) of all the results by dividing `total` by 9. The mean is saved in the variable called `arithmetic_mean`.

When the program is run it displays the following message:

```
The mean percentage mark is 61.77777777777778
```

Make the following changes to the program:

(i) Round the mean percentage to two decimal places.

When the program is run the output should now look as follows:

```
The mean percentage mark is 61.78
```

(ii) Currently, the code divides the total by 9 to calculate the mean. Modify the code so that it divides the total by the number of elements in the results list, regardless of its size.

When the program is run the output should remain the same:

```
The mean percentage mark is 61.78
```

(iii) Complete the function `get_grade` so that it sets the variable `grade` to the correct grade using the parameter `result` and the information provided in the table below.

Result	Grade
>= 80	Distinction
>= 65	Upper Merit
>= 50	Lower Merit
>= 40	Pass
< 40	Unsuccessful

When the program is run the output should remain the same:

```
The mean percentage mark is 61.78
```

This question continues on the next page.

(iv) Extend the program so that it calls the function `get_grade` and displays the grade in a message such as: *The grade for the average result is [grade]*.
You will need to pass `arithmetic_mean` into the function.

When the program is run the output should now look as follows:

```
The mean percentage mark is 61.78
The grade for the average result is Lower Merit
```

(v) Add code so that the program finds the lowest and highest scores in `results`.
The program should display this information in two separate messages such as:

The lowest score is [lowest score]

The highest score is [highest score]

When the program is run the output should now look as follows:

```
The mean percentage mark is 61.78
The grade for the average result is Lower Merit
The lowest score is 32
The highest score is 88
```

(vi) Add two features to count how many scores in `results` are a) less than 40 and b) between 50 and 79 inclusive. The program should display both counts in two separate messages such as:

The number of scores below 40 is [count1]

The number of scores between 50 and 79 inclusive is [count2]

When the program is run the output should now look as follows:

```
The mean percentage mark is 61.78
The grade for the average result is Lower Merit
The lowest score is 32
The highest score is 88
The number of scores below 40 is 2
The number of scores between 50 and 79 inclusive is 5
```

(vii) Extend the program so that it determines and displays the longest run (sequence) of consecutive result increases.

Example:

Input: `results = [39,32,62,88,51,62,64,81,77]`
Output: `[51,62,64,81]`
Explanation: The sub-list `[51,62,64,81]` is the longest run of result increases. The value 77 breaks the sequence because it is less than its previous value, 81.

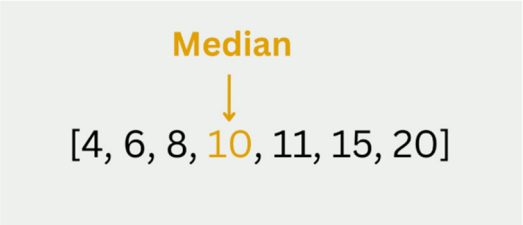
When the program is run the output should now look as follows:

```
The mean percentage mark is 61.78
The grade for the average result is Lower Merit
The lowest score is 32
The highest score is 88
The number of scores below 40 is 2
The number of scores between 50 and 79 inclusive is 5
The longest run of result increases is [51, 62, 64, 81]
```

Save your file using the format **ExaminationNumberQuestion16_A.py**. For example, you would save the file as **123456Question16_A.py** if your Examination Number was 123456.

This question continues on the next page.

(b) Open the program called **Question16_B.py** from your device. This file contains only two comments, on lines 1 and 2.



Before adding any code, you should save your working copy of the file using the format **ExaminationNumberQuestion16_B.py**.

For example, you would save the file as **123456Question16_B.py** if your Examination Number was 123456.

Enter your Examination Number in the space provided on **line 2** in your Python file**.**

The median of a list of values is the middle value in that list after it has been sorted in either ascending or descending order. If the number of values in the list is odd the median is the middle value. However, if the number of values in the list is even the median is found by calculating the mean of the two middle values i.e. by adding both middle values and then dividing the result by two.

Write a Python program to find the median of a list of zero or more values.

You should use comments throughout your program to explain your code. You may wish to reuse some of the code you used in **part (a)** as part of your solution.

Your program should meet the following requirements:

- Initialise a list of integers.
- Display the list.
- Sort the list.
- Display the sorted list.
- Determine the median by examining the list length.
    - **Odd**: The number of elements divided by 2 will have a remainder of 1. In this case the median is the element at the middle position of the sorted list.
    - **Even**: The number of elements divided by 2 will have no remainder. In this case the median is the mean of the two middle elements in the sorted list.
- Display the median.
- Display an error message if the list is empty.

Note:

1. You may **not** make use of the `statistics.median()`, `numpy.median()` or any other Python library median function in your solution. You must write the code to calculate the median yourself.
2. You should test that your program works with an even and odd number of values.
3. You should test that your program works with an empty list.

Example outputs are shown on the next page.

Sample output 1:

```
The initial list of values is: [27, 13, 32, 50, 16]
The sorted list of values is: [13, 16, 27, 32, 50]

The median is 27
```

Sample output 2:

```
The initial list of values is: [27, 13, 32, 50, 16, 29]
The sorted list of values is: [13, 16, 27, 29, 32, 50]

The median is 28.0
```

Sample output 3:

```
The initial list of values is: []

The list is empty. Cannot compute the median.
```

Use the format **CandidateNumberQuestion16_B.py** to save your file. For example, you would save the file as **123456Question16_B.py** if your candidate number was 123456.

Space for rough work.

This page will not be reviewed by an examiner.

Space for rough work.

This page will not be reviewed by an examiner.

Do not hand this up.

This document will not be returned to the State Examinations Commission.

Leaving Certificate – Higher Level

Computer Science – Section C

Wednesday 21 May
Morning 11:30 – 12:30