2020L219ACES

Coimisiún na Scrúduithe Stáit
State Examinations Commission

Leaving Certificate Examination
Sample Paper

Computer Science

Section C

Higher Level

Time: 1 hour

80 marks

Examination number					

Centre stamp

Instructions

There is one section of the examination paper in this booklet.

Section C	Programming	80 marks	1 question

Answer all parts of the question on your digital device.

Instructions are provided for each question.

Ensure that you save your work regularly and when you complete each question.

Do not change the file names or save your work under different file names.

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.

Section C	Programming	80 marks

Answer all question parts.

Question 16

According to a recent report in the Irish Independent (19 August 2019) one in four children and three out of four adults in Ireland are overweight or obese. This causes significant increases in illnesses such as diabetes and heart disease. To determine whether a person is overweight or obese, a measure called the Body Mass index (BMI) can be used.

BMI is defined as body weight (in kilograms) divided by the square of the body height (in metres). If height is given instead in centimetres, then the formula for calculating BMI is:

$$BMI = \frac{weight}{height^2} \times 10000$$

(a) Open the program called **Question16_A.py** from your device.
Enter your Examination Number in the space provided on **Line 2.**

```
1  # Question 16(a)
2  # Examination Number:
3
4  weight = int(input("Enter weight (in kilograms): "))
5  height = 180 # centimetres
6
7  bmi = weight / (height * height) * 10000
8
9  print("BMI is: ", bmi)
10
```

This program is designed to calculate and display the BMI for a person whose weight is entered by the end-user and whose height is 180 cm.

A sample run of the program is displayed below – the user enters a value of 90 (kilograms) for **`weight`** and the program displays the resulting BMI.

```
Enter weight (in kilograms): 90
BMI is: 27.777777777777778
```

Modify the program to do the following:

(i) Insert a comment to say '*read weight*' in the appropriate location in the program to show where the weight is input.

(ii) Currently in the program the value of the variable **`height`** is hard-coded to 180. Modify the program so that it prompts the user to enter a value for **`height`**. The value should be converted to an integer.

When the program is run the output may look as follows:

```
Enter weight (in kilograms): 90
Enter height (in centimetres): 180
BMI is: 27.77777777777778
```

(iii) By using the function **`round`**, or otherwise, modify the program so that the value of the BMI displayed is rounded to one decimal place.

When the program is run the output may look as follows:

```
Enter weight (in kilograms): 90
Enter height (in centimetres): 180
BMI is: 27.8
```

(iv) By using the function **`pow`**, or otherwise, replace the expression (**`height * height`**) with an alternative, but equivalent, implementation.

(v) Incorporate the following function definition into your program and insert a line so that the function is called before the user enters any data.

```
def display_intro():
    print("Welcome to my BMI calculator!")
```

When the program is run the output may look as follows:

```
Welcome to my BMI calculator!
Enter weight (in kilograms): 90
Enter height (in centimetres): 180
BMI is: 27.8
```

(vi) A BMI number alone might not mean much to the average person. A more meaningful output might be one of the four BMI categories: underweight, normal weight, overweight or obese.

Extend the program so that it displays the BMI category according to the BMI ranges in the table below.

BMI Category	BMI Range
Underweight	less than 18.5
Normal	from 18.5 to 24.9
Overweight	from 25 to 29.9
Obese	greater than or equal to 30

The table below shows the expected BMI values and categories for some sample weight and height values. You could use this data to test your program.

`weight` (kgs)	`height` (cms)	`bmi` (kg/cm^2)	BMI Category
50	165	18.3	Underweight
70	170	24.2	Normal
90	180	27.8	Overweight
90	170	31.1	Obese

When the program is run the output may look as follows:

```
Welcome to my BMI calculator!
Enter weight (in kilograms): 78
Enter height (in centimetres): 180
BMI is:  24.1
Normal
```

Save and close your file before moving on to the next part.

(b) Open the program called **Question16_B.py** from your device.
Enter your Examination Number in the space provided on **Line 2.**

```
1  # Question 16(b)
2  # Examination Number:
3
4  # For this question it is useful to understand ...
5  # 1. randint(a, b) returns a random integer N such that a<=N<=b.
6  # 2. s.append(x) appends the element x to the end of list s.
7
8  from random import *
9
10 heights = [] # an empty list of heights
11 weights = [] # an empty list of weights
12
13 # Loop to build up the lists with random values
14 for count in range(10):
15     # a random integer between 150 and 210
16     heights.append(randint(150, 210))
17     # a random integer between 50 and 130
18     weights.append(randint(50, 130))
19
20 # Display the lists
```

The program creates two empty lists – **`heights`** and **`weights`**.

It then uses a **`for`** loop to populate both lists with 10 random values as follows:
- each element in **`heights`** is a random integer between 150 and 210 (cm)
- each element in **`weights`** is a random integer between 50 and 130 (kg)

Modify the program to do the following:

(i) Display the contents of both lists, along with the labels **Heights** and **Weights**.

When the program is run the output may look as follows:

```
Heights: [202, 180, 200, 152, 159, 165, 206, 208, 157, 154]
Weights: [87, 118, 106, 53, 96, 106, 87, 121, 129, 95]
```

(ii) Currently the program generates 10 pairs of values every time it is run. Modify the program so that it generates a variable number of values entered by the user.

When the program is run the output may look as follows:

```
Enter the number of pairs of values you wish to generate: 7
Heights: [151, 198, 168, 197, 154, 197, 166]
Weights: [120, 95, 111, 89, 72, 87, 115]
```

(iii) Extend the program to create and populate (using a separate loop) a third list called **`bmi_values`**.

Each element of **`bmi_values`** should be calculated using the corresponding values from **`heights`** and **`weights`** (that is **`bmi_values[i]`** should be calculated using **`heights[i]`** and **`weights[i]`**).

You may wish to use the code from **part (a)** to calculate the BMI value.

The BMI values should be rounded to one decimal place.

When the program is run the output may look as follows:

```
Enter the number of pairs of values you wish to generate: 7
Heights: [178, 199, 160, 205, 203, 153, 166]
Weights: [106, 119, 59, 53, 92, 109, 105]
BMI values: [33.5, 30.0, 23.0, 12.6, 22.3, 46.6, 38.1]
```

Save and close your file before moving on to the next part.

(c) Open the program called **Question16_C.py** from your device.
Enter your Examination Number in the space provided on **Line 2.**

```
1  # Question 16(c)
2  # Examination Number:
3
4  bmi_values = [24, 19, 33, 35, 27, 18, 15, 33, 35, 23, 32, 23]
```

The program initialises a list called **bmi_values** with 12 integer values.

Write a Python program to do the following:

(i) Determine and display the number of obese values in the list.
(Recall from **part (a)** that if the BMI is greater than or equal to 30 then it is categorised as obese.)

When the program is run the output may look as follows:

```
Obese: 5
```

(ii) Determine and display the largest and second largest values in the list.
(Note that the list contains values that are repeated.)

When the program is run the output may look as follows:

```
Largest: 35
Second Largest: 33
```

(iii) Implement a general function that returns the n^{th} largest value from the list.

Name the function **find_nth_largest** as shown in the function definition header:

```
def find_nth_largest(n, list_of_values):
```

The function accepts two parameters, **n** and **list_of_values**. The function body should be coded to determine and return the n^{th} largest value in the list.

An example of the function being used to determine the 3^{rd} largest value of the list is shown.

```
print(find_nth_largest(3, bmi_values))
```

When the program is run the output may look as follows:

```
32
```

Save your file.
Ensure that you have saved and closed all files before you finish the examination.

There is no examination material on this page

There is no examination material on this page

There is no examination material on this page

Acknowledgements
None

Leaving Certificate – Higher Level

Computer Science – Section C

Sample Paper
Time: 1 hour