IT1007 Introduction to Programming with Python and C
Lab Exercise 06

Submission instructions:

1. There are two parts in this lab. For Part A, you have to submit within the same day of your lab session. For your Part B, you have six days to work on it. (E.g. if your TLab is on Monday, then your deadline isthe coming Sunday midnight.)
2. Complete your code using the skeleton files provided, then**test your code on your computer first** before submitting to Coursemology.
3. To submit your code on Coursemology, click on "Labs" in the sidebar followed by the appropriate"Attempt" button.
4. **Copy ONLY the required function** from your completed skeleton file into the Coursemology codewindow.
5. Click "Run Code" to test that your function works on Coursemology.
6. Click "Finalise Submission" to submit your code for the **ENTIRE Part A/B**. **You will not be able to amend your code after you have finalised your submission.**
7. You must name your functions exactly as the questions state.

Failure to follow each of the instruction will result in 10% deduction of your marks.

Part A (Deadline: Same day of your TLab)

Questions 1Counting days (10 Marks)

Define a function ComputeDays which reads in two inputs: date (1-31) and month (1-12). You may assume that all inputs are valid. The program calculates the number of days since 1st January 2017 till that date. Do not change the main function provided.

Sample runs:

```
Enter date and month: 1 1
Day 1.

Enter date and month: 15 8
Day 227.

Enter date and month: 31 12
Day 365.
```

Question 2: Parity finder [20 Marks]

Declare a global integer array DATA that can store 20 integers. Write the following functions and call from your main function appropriately.

(i) [5 Marks] A random number generator that generates integers 0 and 1; Use this function to generate and store the numbers generated in the array and then pass the array to MYPRINT(…) to print the array;

(ii) [5 Marks] A function MYPRINT(…) that prints the array whenever it is called;

(iii) [10 Marks] A function PARITY(..) that adds all the numbers of your array DATA and print the result from this function; Display the results as: "Array is of ODD PARITY type" if the result is a odd number or "Array is of EVEN PARITY type" if the result is an even number;

Sample output:

```
[ 0 1 1 0 1 0 1 1 0 1 0 1 1 0 1 1 0 0 0 0 ]          /* MYPRINT(DATA)*/
Array if of EVEN Parity type                         /*PARITY(DATA)*/
```

Part B (Deadline: Day of TLab + 6 days)

Question 1:Signal Analysis [70 marks]

A slow time-varying random signal is sampled and the sampled values are stored in an array for processing. Write a program that does the following functions.

General instructions: First read the entire problem carefully before you begin designing your solution. You may consider declaring all the arrays as global arrays. Clearly define function prototypes. You may consider writing the functions as separate files or in the same main file itself. Clearly write comments for your code.

Important Note: For meaningful and correct comments for the entire problem solution, **5 Marks** (on the whole) will be awarded.

This question is all about random generator, so the value of your array will be different from the sample output. So the sample output is only for you to know the format.

Consider a floating-point array **Signal**[] that can store **100** numbers in the range **-10 to +10**.

(i) [**Signal Generation (5 Marks)** Use the given code at the end of the problem statement on Page 4 to generate these float-point numbers randomly and store in the Signal[] array. Print this Signal[] array first; This is your **base sampled signal**.

The code on page 4 is float-point generator. You need to modify it to define a function generate_Signal() to store the numbers into Signal[], then define a print_signal(float arr[], int size) to print the array out.

Sample output:

[-7.12,7.77,…,-6.91]

(ii) [**Max & Min Search (10 Marks)**] Pass this Signal[] array to a function **Max_Min()** that searches for maximum and minimum values from this array of 100 integers; Print your output from this Max_Min() function; If there are more than one maximum and minimum values it is sufficient if you print just once the values.

Sample output:

```
Max: 9.959951
Min: -9.977858
```

(iii) [**Filtering (20 Marks)**] Pass the original array Signal[] to a **Zero_Bias()** function which replaces all the negative values to 0; Save this array as a **Zero_Bias_Array[];** Print this new array.

(iv) [**Zero Count (15 Marks)**Pass this above Zero_Bias_Array[] to another function **Zero_Count()** which counts the number of zeros and the indices where they occur in the list and **print** your result from this Zero_count() function itself.

Sample output:

```
Zero at index 1
Zero at index 2
Zero at index 4
Zero at index 6
Zero at index 9
Zero at index 10
Zero at index 11
……
Total zero count = 53
```

(v) [**Threshold Detection (15 Marks)**] Pass the Zero_Bias_Array[] to a **Threshold_Detect()** function to detect how many times the signal values exceed (strictly greater than) 4.

Sample output:

```
Threshold count = 31
```

```
#include <stdio.h>

#include <stdlib.h>

#include <time.h>


int main()

{

   int toss;

   srand((unsigned int)time(NULL));

   /* Generate random numbers in the range [-10,10] */


   for (int i=0;i<20;i++){

      toss = rand()%2; /* flip a coin */

      if(toss == 0)/* -ve num& a 0 */

        printf("%f\n", (float)rand()*-10/(float)(RAND_MAX));

  else/* +ve num& a 0*/

        printf("%f\n", (float)rand()*10/(float)(RAND_MAX));

   }

   return 0;

}
```