Lab Assignment Week 1 – Warm up Exercises

Instructions:

- You can use any programming language platform (ex. C, C++, Java or Python) for these Competitive Programming problems
- Use only built-in functions in these languages as the installation of external libraries is not allowed in online competitions.
- Solve as many exercises as you can. Implementing multiple solutions and comparing their feasibility is highly encouraged.
- Students who complete assignments earlier during the labs can continue doing lab by solving online coding exercises from CodeChef, HackerRank etc. Lab instructors will grade them with bonus marks.

Exercise 1:

You have to count the numbers from [Left, Right] that contains any of the given digits at odd locations.

For example: Left = 10, Right = 20, Digits = [1, 2, 3, 7] then numbers between 10 and 20 containing 1, 2, 3, 7 at odd are: 11, 12, 13, 17. So output should be 4. Number 10 cannot be included because 10 contains 1 at even location.

Tips: Try multiple solutions using numbers, string and compare their memory and running time.

Exercise 2:

There is a counter in a bank after lunch time where people with following tokens are standing in a queue [20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]. Suddenly 3 more new counters are opened and the current one is closed. So, people at the back of original counter got out easily and start rushing towards other counters. Now they form three new lines in the order: new counter 1 [30, 27, 24, 21], new counter 2 [29, 26, 23, 20], new counter 3 [28, 25, 22]. The number of new counters may vary. If number of new counters is 2, two counters will have tokens then in the order: new counter 1 [30, 28, 26, 24, 22, 20], new counter 2 [29, 27, 25, 23, 21]. So, you need to get the number of new counters during execution time.

Tips: You need to find the suitable data structure for this problem. That data structure can be implemented from scratch or using built-in functions. Try implementing both version and compare their memory and running time.

Exercise 3:

Given a string “BennettUniversity”, convert every n^{th} location into upper case. If n=1 then converted string is “BENNETTUNIVERSITY”, if n=2 then “BEnNeTtUnIvErSiTy”, if n=3 then “BeNneTtUNivErsIty”. If a character is uppercase already then it can be left as it is. The value of n is gathered during executing time.

Tips: Use different methods to convert cases in strings. It just needs some “bit of Manipulation”. Finding more variety of solutions is better.

Exercise 4:

There is a dog standing on an array from [-∞,…-4, -3, -2, -1, 0, 1, 2 ,3, 4…+∞]. The dog initially stands on a positive number of the array say i=10. If you tell the dog to jump with steps s = 5 then the dog jump towards the negative side say [10, 5, 0] once it touches the 0 or negative number it comes again towards the original local 10, so now it will jump from [0,5,10]. If you say s=7 then jumps will be [7,2,-5,2,7]. Implement this jumping dog without using any for loop or while/do while loop.

Exercise 5

There is a family with 10 members of different ages. They are sitting in the order from younger to older [2, 5, 10, 13, 23, 30, 32, 40, 60, 70]. Now given a random age *x* = 20 between 2 and 70, you need to find index of the member close to the random guess x = 20. The answer is 5^{th} member with age 23. List has to be predefined. Random integer guess x should be taken during running time.

Tips: Implement different approaches and compare their time complexities.

Submissions:

Submit your assignment as single Zip file to LMS before **Sunday 12 Jan 11.59 PM**

You can zip all the programs into a file named as roll_number_CP_w1.rar (example e19cse001_CP_w1.rar). Individual programs can be named as exercise1.py, exercise2.py etc. Multiple solutions for same exercise can be named as exercise1_a.py, exercise1_b.py, exercise1_c.py etc.