Data Structures and Algorithms Lab
SE-F22
LAB-02

Issue Date: February 2, 2024

Start Time: 8:45AM
Submission Time: 11:45AM

Total Marks: 60

The objective of this lab is to:
Determine time complexity, growth functions & big oh notation of code snippets. Practice writing code in O(n) time complexity.

Instructions:
1) Follow the question instructions very carefully, no changes in function prototypes are allowed.
2) You could solve the following growth functions on paper.
3) Anyone caught in an act of plagiarism would be awarded an "F" grade in this Lab.

Task 01(Time Complexity Analysis) [20 Marks]

Consider the following code snippets. You are required to write the growth function and specify the Big-Oh of each of the following codes:

1
```
int sum = 0;
for (i = 0; i < n; i++)
{
        for (j = 0; j < n * n; j++)
        {
                sum++;
        }
}
```

2
```
int sum = 0;
for (i = 1; i < n; i *= 2)
{
       for (j = 1; j < n; j++)
       {
              sum++;
       }
}
```

3
```
int sum = 0;
for (int j = 10; j < n/2; j += 2)
{
       if (j % 2 == 0)
       {
              for (int k = 0; k < n; k++)
              {
                     sum++;
              }
       }
}
```

4
```
int unknown(int n)
{
        int i, j, k = 0;
        for (i = n / 2; i <= n; i++)
                for (j = 2; j <= n; j = j * 2)
                        k = k + n / 2;
        return k;
}
```

Task 02 (findSqrt) [10 Marks]

You are provided an integer ‘n’. Your task is to find the square root of that integer. If the square root is not a perfect square, then return the floor value of the sqrt(n).

Example:

Input: 28
Result: 5
Explanation: The square root of ‘28’ is ‘5.292’. So, the floor value of ‘5.292’ is 5.

Note: In order to get full marks, write code that has time complexity better than O(n).

Task 03 (Product Except Self) [15 Marks]

You are given an integer array ‘nums’ of any size. Your task is to return another array let’s say ‘result’ such that *result[i]* is equal to product of every element of *nums* except *nums[i]*.

Example:

Input: {1,2,3,4}
Output: {24,12,8,6}
Explanation: The value at index 1 of output array is calculated as the product of all elements of input array expect the value at index 1 i.e. 1*3*4 = 12.

Note:

Find solution in linear time complexity i.e O(n).

Function Prototype:

```
int * productExceptSelf(int * nums, int size)
```

Task 04 (Average of subarray) [15 Marks]

You are given an integer array ‘nums’ and an integer value ‘k’. You are required to find the average of each subarray of size ‘k’ and store that to a double array. In the end return that double array.

Example:

Input: {1,12,-5,-6,50,3}, k =4
Output: {0.5,12.75,10.5}
Explanation: The first subarray of size 4 is 1,12,-5,-6 and average of these is 0.5, then the next subarray is 12,-5,-6,50 and its average is 12.75 & final subarray is -5,-6,50,3 and its average is 10.5.

Note:

Find solution in linear time complexity i.e O(n). Assume the subarray is continuous.

Function Prototype:

```
double * findAverage(int * nums, int size, int k)
```

Good Luck!
