CS 3304 – Data and Information Structures
Assignment 3 & 4

Due: 04/02/2023, 23:59

Submission: Submit your solutions as a single **.zip** file through Blackboard. **.zip** file will contain

- (Q1) Class file for linked stack and a driver file.

- (Q2**)** One cpp file that contains two functions and the main function.

- (Q3-9**)** One pdf file that contains your solutions.

Q1 (40 pts). Write a linked stack class as a template class. You can use the linked stack class given in your textbook (Figure 7.11 and Figure 7.12). You can find the code on Blackboard under "Course Content / Code Samples / Chapter 7". You only need to convert this class to a template. You can keep all the data and function components.

Write a program that reads a file named "input.txt", reads the numbers in the line and add them into two stacks with the following rules:

- If the number is odd:
  - Push into a stack of odd numbers if the number is not equal to the top of the stack.
  - Pop the top of the stack if the number is equal to the top of the stack.
- If the number is even:
  - Push into a stack of even numbers if the number is not equal to the top of the stack.
  - Pop the top of the stack if the number is equal to the top of the stack.

The program should print the top of the both stacks for each line of the file. You need to read the file line by line and add\remove numbers into appropriate stack. You can assume that the file only has numbers separated by a space. You will print your output to the console for each line.

An example "input.txt" and console output is given below:

input.txt

```
3 5 8 6 7 8 7
4 0 3 6 9 2
9 7 2 6 7 9
8 3 2 12 12 3 2 9 8
13 18 21 18 16 14 14 21
```

Output

```
Odd Top: 5          Even Top: 8
Odd Top: 9          Even Top: 2
Odd Top: NULL       Even Top: 6
Odd Top: 9          Even Top: NULL
Odd Top: 13         Even Top: 16
```

Q2 (20 pts). Write the following two functions and call those functions in the main function to print the result.

a. (10 pts) Write a recursive C++ function named **recursiveSumPositiveEven** that accepts an integer array as a parameter and returns the sum of the positive even numbers in the array using recursion without using iteration. For instance, **recursiveSumPositiveEven(a)** will return 8 if array a contains {1,0,-4,2,6}. **Hint:** the sum of the positive even numbers is equal to "first element + the sum of the positive even numbers starting from the second element" if the first element is positive and even. Otherwise, it is equal to "the sum of the positive even numbers starting from the second element".

b. (10 pts) Write a recursive C++ function named **countDigits** that accepts a String parameter and returns the number of digits using recursion without using iteration. For instance, **countDigits("Abc12.@0K")** will return 3.

Q3 (40 pts).

a. (10 pts) Prove that if $T(n) = 5n^2 + 3n + 10$, then $T(n) = O(n^2)$. Hint: You need to find two positive constant values c and n_0 such that $T(n) \leq cn^2$ for all $n \geq n_0$.

b. (10 pts) Suppose the running time of an algorithm is $7n^2 + 22m^2 + 4nm + 10m + 200$, where n and m are positive integers such that $n \geq m$. Prove that this algorithm is $O(n^2)$.

c. (20 pts) Give an analysis of the running time (Big-Oh notation) for the following algorithms:

```
sum = 0;
i = 0;
while(i < n){
        sum+=i;
        i+=3;
}
while(i > 0){
        sum-=i;
        i-=2;
}
while(i < sqrt(n)*2){
        sum+=i;
        i++;
}
i = 3*n;
while(i > 0){
        sum-=i;
        i--;
}
print(sum);
```

```
sum = 0;
i = 0;
while (i < n) {
        j = 1;
        while (j <= n) {
                sum += (i + j);
                j++;
        }
        i*= 2;
}
i = 0;
while (i < 10) {
        j = 1;
        while (j <= n) {
                sum -= (i + j);
                j+=2;
        }
        i++;
}
i = 0;
while (i < 100000) {
        sum += i;
        i++;
}
print(sum);
```

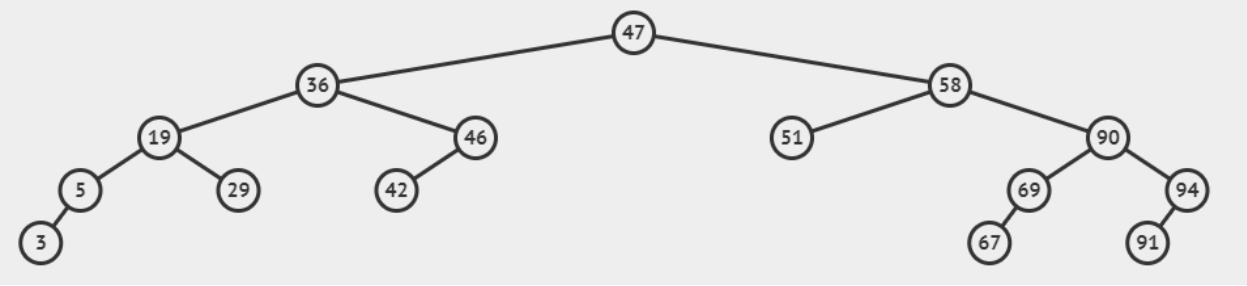
Firstly calculate T(n) and then find a tight upper bound (Big-Oh) for each algorithm.

Q4 (10 pts). Using the hash function H(x) = 3*x + 2 (mod 13), and linear probing, show the hash table that results when the following integers are inserted in the order given:

- 26, 42, 5, 44, 92, 38, 40, 45, 12, 57, 80, 31.

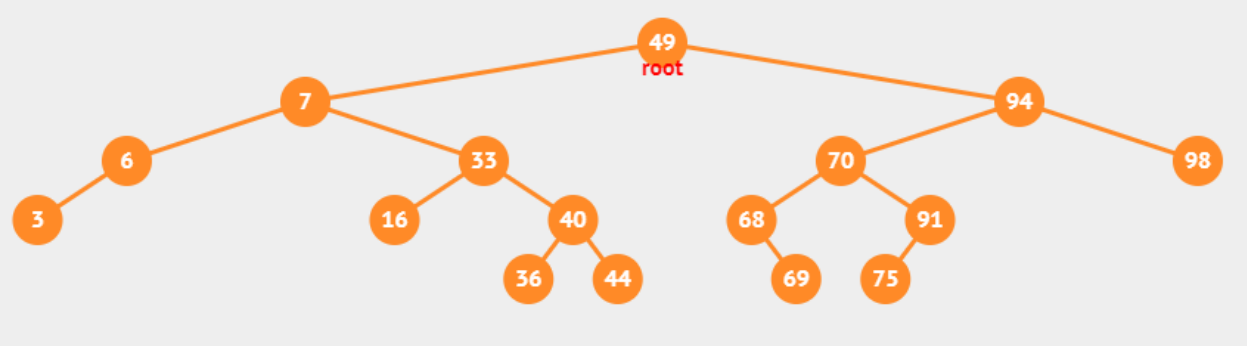
Q5 (20 pts). Given the following binary search tree (BST), show the BST after each insertion/deletion given below:

- Insert 48
- Insert 75
- Insert 4
- Insert 22
- Delete 58
- Delete 47
- Delete 5
- Delete 19



Q6 (15 pts). Given the following binary tree, show the order of the elements if

a. Inorder traversal is performed.
b. Preorder traversal is performed.
c. Postorder traversal is performed.



Q7 (20 pts). Given the array of numbers, show the elements of the array after each iteration of selection sort, bubble sort, and insertion sort. The algorithms will sort the array in increasing order.

x[1]	x[2]	x[3]	x[4]	x[5]	x[6]
10	7	9	3	8	5

For instance, after the first iteration of the bubble sort, the array will have the following elements:

x[1]	x[2]	x[3]	x[4]	x[5]	x[6]
7	10	9	3	8	5

Q8 (15 pts). Given the array of numbers, show the elements of the array after each iteration of **heapify** operation.

x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]	x[8]	x[9]	x[10]
1	4	5	3	2	8	9	6	7	10

Q9 (20 pts). Given an empty heap (max-heap), show the heap after each insertion/deletion given below:

- Insert 10
- Insert 12
- Insert 13
- Insert 7
- Insert 16
- Insert 19
- Insert 9
- Insert 14
- Delete 19
- Delete 16