CSE 4304-Data Structures Lab. Winter 2022-23
Date: 14 November 2023
Target Group: SWE B
Topic: DSU & Graph Basic
Instructions:

- Regardless of when you finish the tasks in the lab, you have to submit the solutions in the Google Classroom. The deadline will always be at 11.59 PM of the day in which the lab has taken place.
- Task naming format: <fullID>_<Task><Lab><Group>.c/cpp. Example: 170041034_T01L02A.cpp
- If you find any issues in the problem description/test cases, comment in the google classroom.
- If you find any test case that is tricky that I didn't include but others might forget to handle, please comment! I'll be happy to add.
- Use appropriate comments in your code. This will help you to easily recall the solution in the future.
- Obtained marks will vary based on the efficiency of the solution.
- Do not use the <bits/stdc++.h> library.

Task:1

Disneyland has built its airport. The airport has only one runway, which results in heavy traffic. So the authority has decided to create a ***'Runway reservation system'*** for their only runway, which will take the reservation of any transport desired to use the runway.

Before making the entry, the system checks for reservations within the three-minute range of any existing reservation(s). For example, if there is a reservation in the k^{th} minute, it won't take any reservation in k-1, k-2, k-3, k+1, k+2, $(k+3)^{th}$ minutes.

Your task is to help them build the system using Binary Search Trees(BST). (Take reservations until the user gives '-1' as input.)
For every reservation, print the existing reservations in a sorted manner.

Sample Input	**Sample Output**
50 75 53 25 60 29 45 42 28 10 -1	50 50 75 50 75 (Reservation failed) 25 50 75 25 50 60 75 25 29 50 60 75 25 29 45 50 60 75 25 29 45 50 60 75 (Reservation failed) 25 29 45 50 60 75 (Reservation failed) 10 25 29 45 50 60 75

Task:2

'Runway reservation system' has a new requirement. They want to introduce a feature that will allow any transport owner to make a new query, which will allow any transport owner to give a timestamp as input. The system will tell *'How many reservations are in the system before it?'*.

One of their employees proposed a solution that traverses the tree in an In-order fashion and then finds the timestamps that are less than the query. They are not happy with this ***O(n)*** solution. They want you to solve this problem in ***O(height)*** time. Your task is to fulfil their requirement.

The first line of input will give you the number of queries.
Each query gives you the timestamp of a specific reservation. Your task is to find the number for reservation before that timestamp.

Sample input	**Sample output**
(current reservations) 50 75 25 29 45 60 10 -1	
5 45 75 50 10 29	 3 6 4 0 2

Explanation:

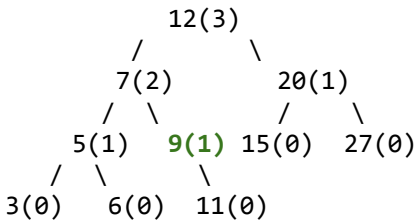
45 has 3 before it (10 25 29)
75 has 6 before it (10 25 29 45 50 60)
50 has 4 before it (10 25 29 45)

Task:3

Suppose a set of numbers is stored in a Balanced BST. Your task is to find the second largest element.

Input will be a sequence of numbers where the last value will be -1, denoting the end. The output will contain two lines. The first line will show the second largest element, and the next line will show the numbers in sorted fashion. Beside each number, show the height of the BST.

Sample Input	**Sample Output**
12 9 5 11 20 15 7 3 6 27 -1	20 Status: 3(0) 5(1) 6(0) 7(2) **9**(1) 11(0) 12(3) 15(0) 20(1) 27(0)
10 15 20 5 8 -1	15 Status: 5(0) 8(1) 10(0) 15(2) 20(0)



Task:4

A set of numbers is stored in a Balanced BST. Given two keys, your task is to print the path between the two nodes and the distance (number of nodes comprising that path).

At first, the numbers will be given as input. Then there will be a series of queries. Each query contains two numbers x & y (x<y). Print the path to reach ‘y’ starting from ‘x’. Then count the number of nodes in that path (include x,y).

Sample Input	**Sample Output**
12 9 5 11 20 15 7 3 6 27 -1	Status: 3(0) 5(1) 6(0) 7(2) **9**(1) 11(0) 12(3) 15(0) 20(1) 27(0)
5 11	5 7 9 11 4
3 12	3 5 7 12 4
6 11	6 5 7 9 11 5
7 9	7 9 2
7 11	7 9 11 3
3 15	3 5 7 12 20 15 6