**CSE 4304-Data Structures Lab. Winter 2021**
**Lab-05**
**Date**: June 26, 2021 (Saturday)
**Target Group:** All Lab groups
**Topic**: Trees, Binary Search Trees

**Instructions**:

- Task naming format: fullID_T01L05_1A.c/cpp
- If you find any issues in problem description/test cases, comment in the google classroom.
- If you find any tricky test case which I didn't include and others might forget to handle, please comment! I'll be happy to add.
- Modified sections will be marked with BLUE colour.

**Task 1:**

Disneyland has built its airport. The traffic for transportation is very high. But unfortunately, they have only one runway. So they have decided to create a '***Runway reservation system***' for their only runway which will take the reservation of any transport desired to use the runway.

However, before making the reservation, the system checks if there is already a reservation within 3 minutes range of any existing reservation. 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<br>75<br>53<br>25<br>60<br>29<br>45<br>42<br>28<br>10<br>-1 | 50<br>50 75<br>50 75 (Reservation failed)<br>25 50 75<br>25 50 60 75<br>25 29 50 60 75<br>25 29 45 50 60 75<br>25 29 45 50 60 75 (Reservation failed)<br>25 29 45 50 60 75 (Reservation failed)<br>10 25 29 45 50 60 75 |

**Note**:

- Do not use any recursive function for this task.
- Utilize the insertion, inorder traversal function of BST.

**Task 2:**

Perform ‘Level-order traversal’ on the BST tree built in Task-1.

| **Sample Input** | **Sample output** |
|---|---|
| 50 75 25 29 45 60 10 -1 | 50 25 75 10 29 60 45 |

Explanation:

1st level - 50
2nd level - 25 75
3rd level - 10 29 60
4th level - 45

Hint: You might need to use a stack/queue to keep track of the nodes to be visited. (E.g queue<node*> q)

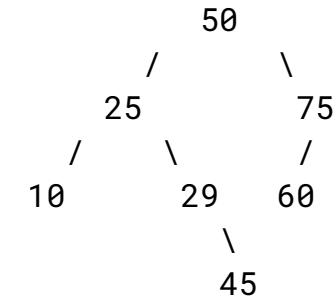
**Task 3**

A great battle is going on between Mike and his enemy team. The General of the enemy team is very clever. He(the General) decided to arrange his soldiers strategically. Firstly, he went to the battlefield and told the soldiers to come one after another. Each soldier(including the General) has a power level.
Whenever a new soldier comes, the General compares the power of the new soldier with his power level. If it is less or equal, that soldier goes to the left of him(the General), otherwise right side. However, the General follows this unique property for every internal soldier(that means, if there is more than one soldier, a new soldier will need to satisfy the property for each of the soldiers including the general).
Now, Mike wants to attack from a random end. Your task is to find out what is the maximum number of soldiers that Mike might need to beat to reach the General of the enemy team.

Each test case starts with the power of the General followed by several soldiers unlit you get -1.

| **Sample input** | **Sample output** |
|---|---|
| 50 25 29 75 10 60 45 -1 | 3 |
| 50 60 70 10 55 -1 | 2 |
| 100 120 110 55 35 45 130 49 -1 | 4 |

Explanation:
For the first case, if the soldiers are organized properly, it will look like




So in the worst case, if mike start killing soldier with power = ‘45’, he’ll need to kill 45, 29, 25 before reaching the General with power = 50

**Task 4:**

'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 certain 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<br>45<br>75<br>50<br>10<br>29 | <br>3<br>6<br>4<br>0<br>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)

[Hint: You might need to use a new attribute for each node called 'subtree-size'. Start traversing from the root and try to use that subtree-sizes wisely. This will lead you to a solution of O(height).]