Data Structures & Applications
Fall 2020
Lab 07 – Trees (Add, Search, Tree Traversals)

Instructor: Saif Hassan **Date: 16th November, 2020**

Instructions:

- At the end of this Lab, you will have to submit all files on LMS.
- Attempt Task01 and Task02 in any environment and Task03 on Hackerrank and submit code here as well.
- File format should be **.zip/.rar** file containing required **.java** files and additional if required.
- File Name should be your **CMSID_Name_Lab07_Section.zip**.
- Create a project named **lab07_dsa** and perform following tasks.
- **.java** files should be as following:
    - **Node.java** --> containing **Node** class only
    - **BinaryTree.java** --> contains complete code for **BinaryTree** Class with implemented functions
    - **Hackerrank.java** --> contains code for **ONLY** four methods such as: Preorder, Postorder, Inorder and Height

Note: Labs submission without following above instructions will not be checked. (No any excuse will be entertained.)

Note: Keep this complete lab code with you till the course ends.

```
class Node {
   Node left;
   Node right;
   int data;

   Node(int data) {
      this.data = data;
      left = null;
      right = null;
   }
}
```

Task 01: (Insertion in Binary Tree)

Binary Tree: A tree whose elements have at most 2 children is called a binary tree. Since each element in a binary tree can have only 2 children, we typically name them the left and right child.

You have been provided above the code for **Node** class, your task is to complete **BinaryTree** Class:

```
1.  class BinaryTree
2.  {
3.      // Root of Binary Tree
4.      Node root;
5.
6.      // Constructors
7.      BinaryTree(int key)
8.      {
9.          root = new Node(key);
10.     }
11.
12.     BinaryTree()
13.     {
14.         root = null;
15.     }
16.
17.     // Methods
18.     public void addData(int data) {
19.
20.         // insert elements in a tree so that left subtree of parent should contain smaller values
21.         // and right sub-tree should contain larger than its parent.
22.         // handle all possible exceptions/errors
23.     }
24.
25.     public boolean searchData(int data) {
26.
27.         // search data from Binary Tree and return true/false, check all possible conditions
28.         // handle all possible exceptions/errors
29.     }
30.
31.     public static void main(String[] args
32.
33.     {
34.         // Test the main method by creating node for different multiple nodes with children
```

```
35.     }
36. }
```

Task 02: (Tree Traversal)

Modify Task 01 and design following methods to access tree elements in different ways

a) Tree: Preorder Traversal
b) Tree: Postorder Traversal
c) Tree: Inorder Traversal
d) Tree: Height of a Binary Tree

Task 03: (Join contest on Hackerrank)

A contest has been created on **hackerrank** website. First signup for the contest and start doing following assigned.

a) Tree: Preorder Traversal
b) Tree: Postorder Traversal
c) Tree: Inorder Traversal
d) Tree: Height of a Binary Tree

Note: Make sure, you code yourself. Anytime during the live class, you might be asked to explain your code.