LAB 09: Binary Search Tree

CS211 – Data Structures and Algorithms
Usman Institute of Technology
Fall 2019

- **How to submit:**
    - Create an account on http://www.turnitin.com/ as a Student (if you don't have already)
    - Use following information at time of sign-up

      CS Section A
        - Class ID: 22664649
        - Enrollment Key: DSFALL19CSA

      CS Section B
        - Class ID: 22664651
        - Enrollment Key: DSFALL19CSB

A. Create a class Node to initialize the left and right pointers in the constructor

1. Add a constructor of the class that takes one argument data in order to set the elements for the tree. The constructor should also initialize left and right node pointers

```
class Node:
    def __init__(self,data):
        // your code goes here
```

B. Create a class BST and implement the following functions.

1. Add a constructor of the class that initializes the root pointer.

```
class BST:
    def __init__(self):
        // your code goes here
```

2. Add a function **Insert()** which inserts a node in the tree recursively.

```
    def Insert(self,value):
        // your code goes here
```

3. Add a function **PreOrder()** which returns a List of elements in the tree.

```
    def PreOrder(self):
        // your code goes here
```

4. Add a function **InOrder()** which returns a List of elements in the tree.

```
def InOrder(self):
    // your code goes here
```

5. Add a function **PostOrder()** which returns a List of elements in the tree.

```
def PostOrder(self):
    // your code goes here
```

6. Add a function **Height()** which returns the height of the tree.

```
def Height(self):
    // your code goes here
```

7. Add a function **FindMin()** which returns the minimum element of the tree.

```
def FindMin(self):
    // your code goes here
```

8. Add a function **FindMax()** which returns the maximum element of the tree.

```
def FindMax(self):
    // your code goes here
```

9. Add a function **Successor()** which returns the successor of an element in the tree. i.e. Minimum value in the right subtree.

```
def Successor(self):
    // your code goes here
```

10. Add a function **Predecessor()** which returns the predecessor of an element in the tree. i.e. Maximum value in the left subtree.

```
def Predecessor(self):
    // your code goes here
```

11. Add a function **Delete()** which deletes a node from the tree.

```
def Delete(self,value):
    // your code goes here
```