Data Structures & Algorithms – Lab #8

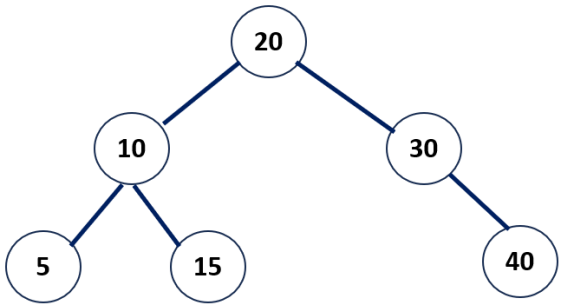
Aim: Getting Familiar with the Binary Search Trees (BST)

Topics:

1. Binary Search Tree (BST)
2. Implementation of BST Node
3. Operations on BST (Search, Insert, Remove)

Lab Questions

Q1 – Implement the BST node, then create the following BST without using any methods.




```
################ Creating Node Class ###############
class Node:
    def __init__(self,data):
        self.data = data
        self.left = None
        self.right = None

###### Creating all Node objects in the BST ######
root = Node(20)
node10 = Node(10)
node5 = Node(5)
node15 = Node(15)
node30 = Node(30)
node40 = Node(40)

### Assigning left and right nodes to each node ###
root.left = node10
root.right = node30

node10.left = node5
node10.right = node15

node30.right = node40
```

Q2 – Define a function for **searching a value in the BST**. If the searched value is found, print "The target node was found!", and if the target is not found, print "The target was not found!".

```
############ Search for a value in BST ##############
def search(node,target):
    if node is None:
        return None
    elif node.data == target:
        return node
    elif target < node.data:
        return search(node.left,target)
    else:
        return search(node.right,target)

############# Calling search Function ###############
if search(root,15) is None:
    print("The target node was NOT found!")
else:
    print("The target node was found!")
```

Q3 – Define a function for **inserting a value in the BST**. Then, once again, create the **BST** from the first question using the insert function. (Define another function to print all values in the BST.)

```
############ Insert a value in the BST ##############
def insert(node,value):
    if node is None:
        return Node(value)
    elif value < node.data:
        node.left = insert(node.left, value)
    elif value > node.data:
        node.right = insert(node.right, value)
    return node

########## Print all values in the BST ############
def printNode(root):
    if root is not None:
        printNode(root.left)
        print(root.data, end="-->")
        printNode(root.right)

############ Calling insert Function ##############
root = Node(20)
insert(root,10)
insert(root,30)
insert(root,40)
insert(root,15)
insert(root,5)

############ Calling printNode Function ##############
printNode(root)
```

Q4 – Define a function for **removing a value from the BST**.

```
############ Remove a value from the BST #############
def remove(node, value):
    if node is None:
        return node

    if value < node.data:
        node.left = remove(node.left, value)
    elif value > node.data:
        node.right = remove(node.right, value)
    else:
        # Removing a leaf node
        if node.left is None and node.right is None:
            node = None
        # Removing a node with only one child
        elif node.left is None:
            node = node.right
        elif node.right is None:
            node = node.left
        # Removing a node with two children
        else:
            successor = findsuccessor(node.right)
            node.data = successor.data
            node.right = remove(node.right, successor.data)

    return node

############ Finding successor of a node ##################
############ Successor : Next Largest Number #############
def findsuccessor(node):
    current=node
    while current.left is not None:
        current = current.left
    return current
```

```
############# Calling remove Function ###############
remove(root,20)
printNode(root)
```