Task 4

Max Team Size 2

Team Members: _____

In this practice, we will implement a Binary Search Tree (BST) data structure. You need to implement two classes.

i) The first class is a *"node"* class that specifies a node in the BST. The node class has three attributes *value*, left_child and right_child. The __init__ function is given below:

```
def __init__(self, key):
    self.left_child = None
    self.right_child = None
    self.value = key
```

This class also implements the following methods:

a. getValue() that returns the value stored in a node.
b. setValue() that assigns a value to a node. This method takes the value of a node as parameter.
c. getLeft() that returns the left_child of a node.
d. setLeft() assigns a value to the left_child to a node passed as parameter to this method.
e. getRight() that returns the right_child of a node.
f. setRight() assigns a value to the right_child to a node passed as parameter to this method.

ii) The second class is a "BST" class. The BST class contains one attributes called *root* that holds the pointer to the root node of the Binary Search Tree. Initially the root points to None. The __init__ function is given below:

```
def __init__(self):
    self.root = None
```

This class implements the following methods also:

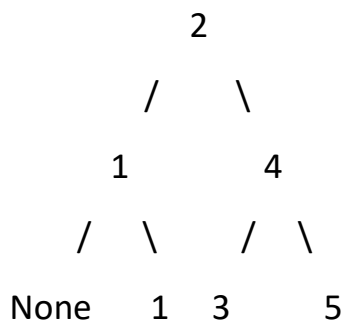
a. searchNode(self, key), that takes the value of a node to be searched in a BST and returns “Found” if the node is in the list else returns “Not Found”.
b. insertNode(self, node), that takes the value of a new node to be inserted in a BST and finds the appropriate position to place this new node.
c. deleteNode(self,key), that takes the value of the key to be removed from the BST and removes the first occurrence of the node with the key value from the BST.

(Hint: Use the algorithm from the lecture slide to implement these methods)

We will use another helper function called inOrder() to print the tree in a particular order (i.e. left-parent-right). This function takes the root of a tree as a parameter.

```
def inorder(node):
    if node: # if the tree is not NULL
        inorder(node.getleft()) # Traverse the left subtree
        print(node.getValue(), end=” ”) # Print the current node
        inorder(node.getright()) # Traverse the right subtree
```

Given the following tree, this function will print the tree as 1 1 2 3 4 5 if called by passing the root of the tree as an argument i.e. inorder(tree.root).



iii) Write a main function that creates the nodes and constructs the tree given above. Then search for keys in the tree e.g. key = 5, as well as keys not in the tree e.g. key = 10. Delete the root from the tree and call the inorder() function to print the tree. The sequence should look like 1 1 3 4 5.