NAME:MUTHUVEAL V

REG NO:3122235002075

UIT2402 -ADVANCE DATA STRUCTURE LAB

EX-1:Implementation of Splay Tree

PYTHON CODE:

```
# splay tree
class Node:
    def __init__(self, key):
        self.key = key
        self.left = None
        self.right = None

def right_rotate(x):
    y = x.left
    x.left = y.right
    y.right = x
    return y

def left_rotate(x):
    y = x.right
    x.right = y.left
    y.left = x
    return y
```

```
def splay(root, key):
    if root is None or root.key == key:
        return root
    if key < root.key:
        if root.left is None:
            return root
        if key < root.left.key:
            root.left.left = splay(root.left.left, key)
            root = right_rotate(root)
        elif key > root.left.key:
            root.left.right = splay(root.left.right, key)
            if root.left.right:
                root.left = left_rotate(root.left)
        return right_rotate(root) if root.left else root
    else:
        if root.right is None:
            return root
        if key > root.right.key:
            root.right.right = splay(root.right.right, key)
            root = left_rotate(root)
        elif key < root.right.key:
            root.right.left = splay(root.right.left, key)
            if root.right.left:
                root.right = right_rotate(root.right)
        return left_rotate(root) if root.right else root
```

```
def insert(root, key):
    if root is None:
        return Node(key)
    root = splay(root, key)
    if root.key == key:
        return root
    new_node = Node(key)
    if key < root.key:
        new_node.right = root
        new_node.left = root.left
        root.left = None
    else:
        new_node.left = root
        new_node.right = root.right
        root.right = None
    return new_node


def delete(root, key):
    if root is None:
        return None
    root = splay(root, key)
    if root.key != key:
        return root
    if root.left is None:
        return root.right
    left_subtree = splay(root.left, key)
    left_subtree.right = root.right
    return left_subtree


def search(root, key):
    root = splay(root, key)
    return root if root and root.key == key else None


def inorder(node):
    if node:
        inorder(node.left)
        print(node.key, end=" ")
        inorder(node.right)


# Main execution
root = None
keys = [100, 50, 200, 40, 60, 150, 300]
for key in keys:
    root = insert(root, key)


print("Inorder traversal after insertion:")
inorder(root)
print()
root = delete(root, 50)
print("Inorder traversal after deleting 50:")
inorder(root)
print()
```

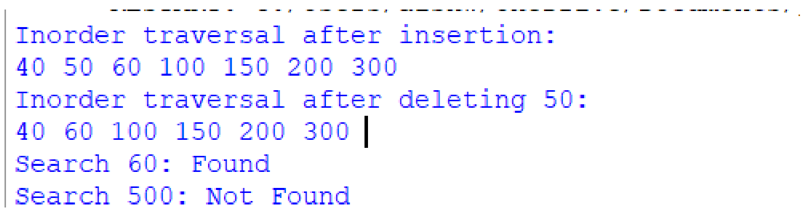
```
found = search(root, 60)

print("Search 60:", "Found" if found else "Not Found")


not_found = search(root, 500)

print("Search 500:", "Found" if not_found else "Not Found")
```

OUTPUT:



```
Inorder traversal after insertion:
40 50 60 100 150 200 300
Inorder traversal after deleting 50:
40 60 100 150 200 300
Search 60: Found
Search 500: Not Found
```