Advanced Data Structures with Python Laboratory (MCAC294)

Assignment – 4

1. Write a Python program to search an element recursively in a binary search tree.

2. Write a Python program to delete a child node from a binary search tree.

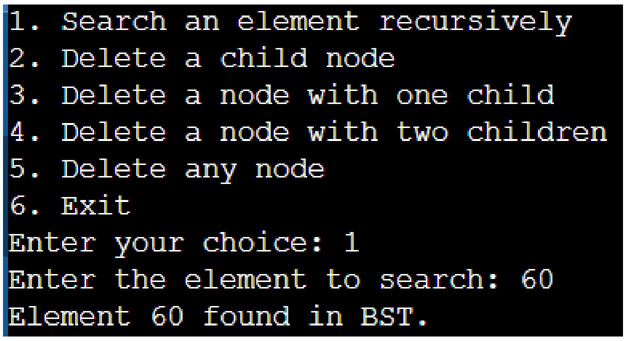
3. Write a Python program to delete a node having one child from a binary search tree.

4. Write a Python program to delete a node having two children from a binary search tree.

5. Write a Python program to delete a node from a binary search tree.

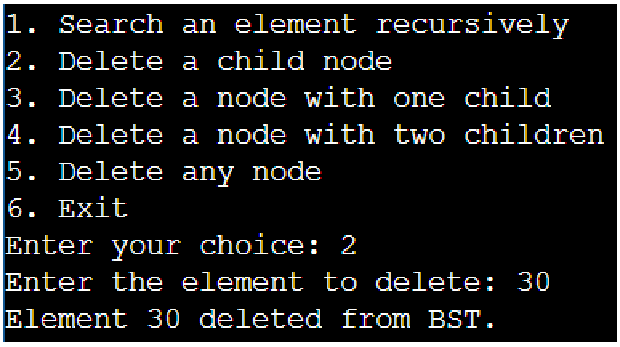
```
class Node:
    def __init__(self, key):
        self.value = key
        self.left = None
        self.right = None
def insert_recursive(root, key):
    if root is None:
        return Node(key)
    if key < root.value:
        root.left = insert_recursive(root.left, key)
    else:
        root.right = insert_recursive(root.right, key)
    return root
def inorder_traversal(root):
    if root:
        inorder_traversal(root.left)
        print(root.value, end=" ")
        inorder_traversal(root.right)

# 1. Search an Element Recursively in BST
def search_recursive(root, key):
    if root is None or root.value == key:
        return root
    if key < root.value:
        return search_recursive(root.left, key)
    return search_recursive(root.right, key)

# 2, 3, 4, 5. Deletion in BST
def delete_node(root, key):
    if root is None:
        return root
    if key < root.value:
        root.left = delete_node(root.left, key)
    elif key > root.value:
        root.right = delete_node(root.right, key)
    else:
        # Case 1: Node has no child (leaf node)
        if root.left is None and root.right is None:
            return None
        # Case 2: Node has one child
        if root.left is None:
            return root.right
        elif root.right is None:
            return root.left
        # Case 3: Node has two children
        successor = root.right
        while successor.left:
            successor = successor.left
        root.value = successor.value
        root.right = delete_node(root.right, successor.value)
    return root
root = None
values = [50, 30, 70, 20, 40, 60, 80]
for val in values:
    root = insert_recursive(root, val)
while True:
    print("1. Search an element recursively")
    print("2. Delete a child node")
    print("3. Delete a node with one child")
    print("4. Delete a node with two children")
    print("5. Delete any node")
    print("6. Exit")
    choice = int(input("Enter your choice: "))
    if choice == 1:
        key = int(input("Enter the element to search: "))
        result = search_recursive(root, key)
        if result:
            print(f"Element {key} found in BST.")
        else:
            print(f"Element {key} not found in BST.")
    elif choice in [2, 3, 4, 5]:
        key = int(input("Enter the element to delete: "))
        if search_recursive(root, key):
            root = delete_node(root, key)
            print(f"Element {key} deleted from BST.")
        else:
            print(f"Element {key} not found in BST.")
    elif choice == 6:
        print("Exiting...")
        break
    else:
        print("Invalid choice! Please try again.")
```

Output for 1:



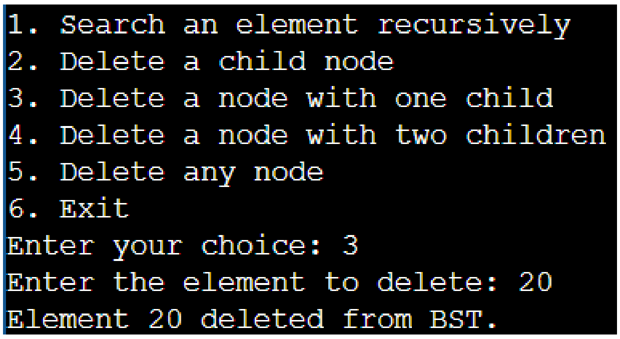
```
1. Search an element recursively
2. Delete a child node
3. Delete a node with one child
4. Delete a node with two children
5. Delete any node
6. Exit
Enter your choice: 1
Enter the element to search: 60
Element 60 found in BST.
```

Output for 2:



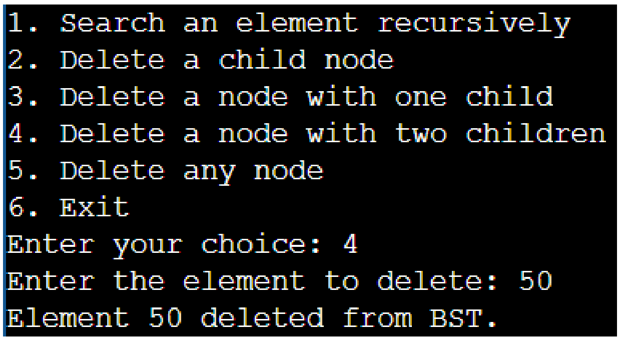
```
1. Search an element recursively
2. Delete a child node
3. Delete a node with one child
4. Delete a node with two children
5. Delete any node
6. Exit
Enter your choice: 2
Enter the element to delete: 30
Element 30 deleted from BST.
```

Output for 3:



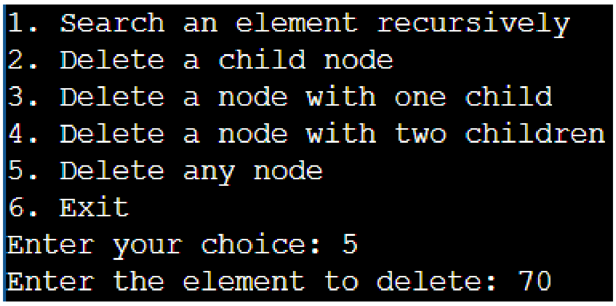
```
1. Search an element recursively
2. Delete a child node
3. Delete a node with one child
4. Delete a node with two children
5. Delete any node
6. Exit
Enter your choice: 3
Enter the element to delete: 20
Element 20 deleted from BST.
```

Output for 4:



```
1. Search an element recursively
2. Delete a child node
3. Delete a node with one child
4. Delete a node with two children
5. Delete any node
6. Exit
Enter your choice: 4
Enter the element to delete: 50
Element 50 deleted from BST.
```

Output for 5:



```
1. Search an element recursively
2. Delete a child node
3. Delete a node with one child
4. Delete a node with two children
5. Delete any node
6. Exit
Enter your choice: 5
Enter the element to delete: 70
```