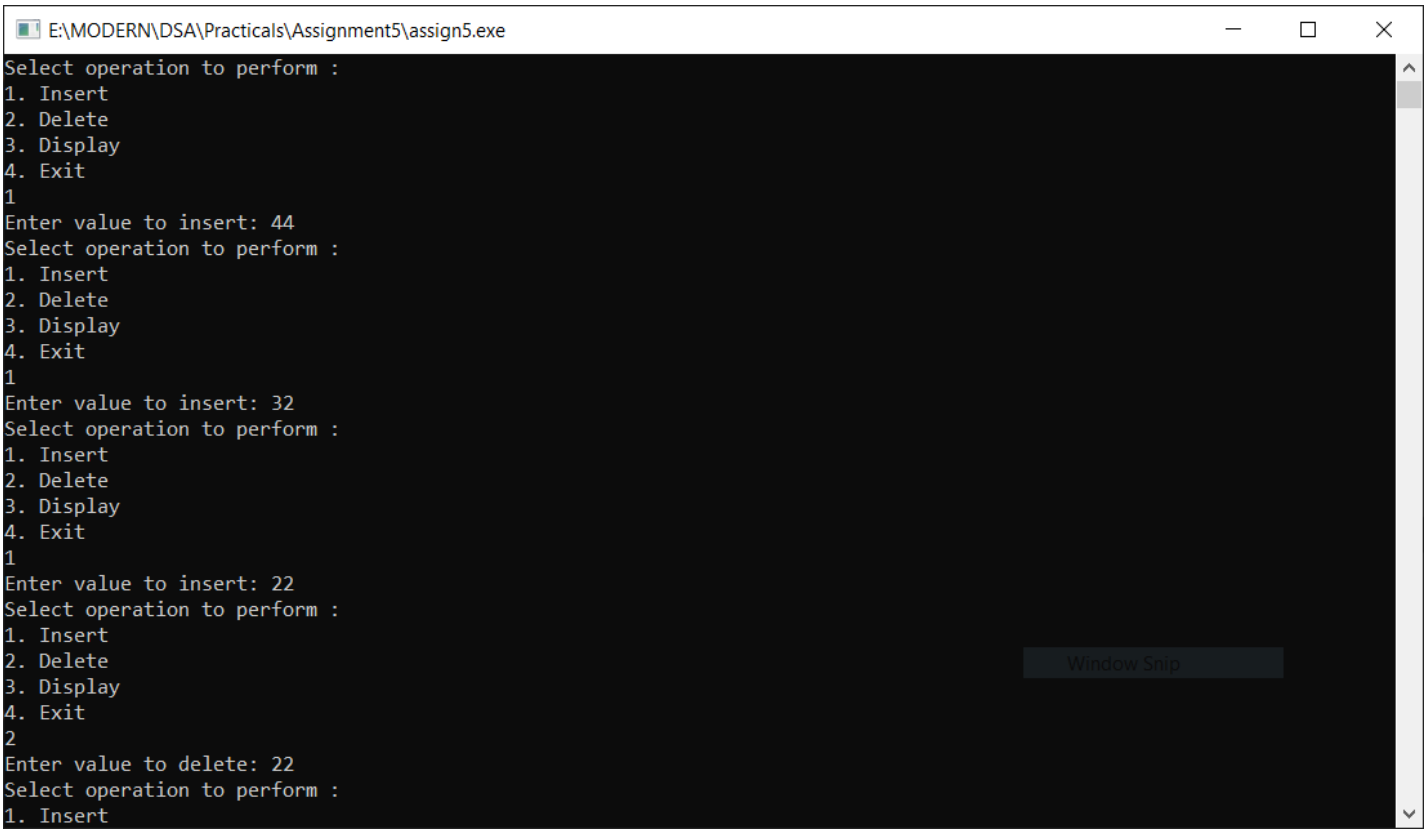
```
 /*
                            ASSIGNMENT NO. 5
NAME- ABRAR SHAIKH                                ROLL NO. - 23570
                            TOPIC- Binary Search Tree
*/


#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node *right;
    Node *left;

    // Constructor to initialize a new node
    Node(int data) {
        this->data = data;
        this->right = NULL;
        this->left = NULL;
    }
};

class BinarysearchTree {
private:
    Node* root;

    Node* insert(Node* node, int data) {
        if (node == NULL) {
            return new Node(data);
        }
        if (data < node->data) {
```

```
            node->left = insert(node->left, data);
        } else {
            node->right = insert(node->right, data);
        }
        return node;
    }

    void shift(Node* node) {
        if (node != NULL) {
            shift(node->left);
            cout << node->data << " ";
            shift(node->right);
        }
    }

    Node* findMin(Node* node) {
        while (node && node->left != NULL) {
            node = node->left;
        }
        return node;
    }

    Node* del(Node* node, int data) {
        if (node == NULL) {
            return NULL;
        }
        if (data < node->data) {
            node->left = del(node->left, data);
        } else if (data > node->data) {
            node->right = del(node->right, data);
```

```
        } else {
            // Node with the data is found
            if (node->left == NULL) {
                Node* temp = node->right;
                delete node;
                return temp;
            } else if (node->right == NULL) {
                Node* temp = node->left;
                delete node;
                return temp;
            }

            Node* temp = findMin(node->right);
            node->data = temp->data;
            node->right = del(node->right, temp->data);
        }
        return node;
    }

public:
    BinarysearchTree() {
        this->root = NULL;
    }

    void insert(int data) {
        root = insert(root, data);
    }

    void display() {
        shift(root);
```

```
        cout << endl;
    }

    void Delete(int data) {
        root = del(root, data);
    }
};

int main() {
    BinarysearchTree tree;
    int choice;

    while (true) {
        cout << "Select operation to perform : \n1. Insert \n2.
Delete \n3. Display \n4. Exit" << endl;
        cin >> choice;

        if (choice == 1) {
            int value;
            cout << "Enter value to insert: ";
            cin >> value;
            tree.insert(value);
        } else if (choice == 2) {
            int value;
            cout << "Enter value to delete: ";
            cin >> value;
            tree.Delete(value);
        } else if (choice == 3) {
            tree.display();
        } else if (choice == 4) {
```

```
            break;
        } else {
            cout << "Enter a valid choice" << endl;
        }
    }

    return 0;
}
```



```
E:\MODERN\DSA\Practicals\Assignment5\assign5.exe
Select operation to perform :
1. Insert
2. Delete
3. Display
4. Exit
2
Enter value to delete: 22
Select operation to perform :
1. Insert
2. Delete
3. Display
4. Exit
3
32 44
Select operation to perform :
1. Insert
2. Delete
3. Display
4. Exit
4

--------------------------------
Process exited after 24.1 seconds with return value 0
Press any key to continue . . .
```

GitHub Repository- https://github.com/abssha/DSA.git