```
 /*
                            ASSIGNMENT NO.6
NAME- ABRAR SHAIKH                                ROLL NO. - 23570
                            TOPIC- Threaded Binary Tree
*/


#include <iostream>
using namespace std;

class Node {
public:
    int data;
    Node *left, *right;
    bool leftThread, rightThread;

    Node(int data) {
        this->data = data;
        left = right = NULL;
        leftThread = rightThread = true;
    }
};

class ThreadedBinaryTree {
private:
    Node* root;

    // Find the leftmost node starting from 'node'
    Node* leftMost(Node* node) {
        if (node == NULL) return NULL;
        while (node->leftThread == false) {
            node = node->left;
        }
```

```
        return node;
    }

public:
    ThreadedBinaryTree() {
        root = NULL;
    }

    // Insert a new node
    void insert(int data) {
        Node* newNode = new Node(data);

        if (root == NULL) {
            root = newNode;
            return;
        }

        Node* curr = root;
        Node* parent = NULL;

        while (curr != NULL) {
            parent = curr;

            if (data < curr->data) {
                if (curr->leftThread == false)
                    curr = curr->left;
                else
                    break;
            } else {
                if (curr->rightThread == false)
```

```
                curr = curr->right;
            else
                break;
        }
    }

    if (data < parent->data) {
        newNode->left = parent->left;
        newNode->right = parent;
        parent->leftThread = false;
        parent->left = newNode;
    } else {
        newNode->right = parent->right;
        newNode->left = parent;
        parent->rightThread = false;
        parent->right = newNode;
    }
}

// In-order traversal
void inorder() {
    Node* curr = leftMost(root);
    while (curr != NULL) {
        cout << curr->data << " ";

        if (curr->rightThread)
            curr = curr->right;
        else
            curr = leftMost(curr->right);
    }
```

```
        cout << endl;
    }

    // Pre-order traversal
    void preorder() {
        Node* curr = root;
        while (curr != NULL) {
            cout << curr->data << " ";

            if (curr->leftThread == false)
                curr = curr->left;
            else {
                while (curr != NULL && curr->rightThread == true) {
                    curr = curr->right;
                }
                if (curr != NULL)
                    curr = curr->right;
            }
        }
        cout << endl;
    }
};

int main() {
    ThreadedBinaryTree tbt;
    int choice;

    while (true) {
        cout << "Select operation to perform: \n1. Insert \n2. In-
order traversal \n3. Pre-order traversal \n4. Exit" << endl;
```

```
        cin >> choice;

        if (choice == 1) {
            int value;
            cout << "Enter value to insert: ";
            cin >> value;
            tbt.insert(value);
        } else if (choice == 2) {
            cout << "In-order traversal: ";
            tbt.inorder();
        } else if (choice == 3) {
            cout << "Pre-order traversal: ";
            tbt.preorder();
        } else if (choice == 4) {
            break;
        } else {
            cout << "Enter a valid choice" << endl;
        }
    }
    return 0;
```

```
E:\MODERN\DSA\Practicals\Assignment6\Assign6.exe
Select operation to perform:
1. Insert
2. In-order traversal
3. Pre-order traversal
4. Exit
1
Enter value to insert: 4
Select operation to perform:
1. Insert
2. In-order traversal
3. Pre-order traversal
4. Exit
1
Enter value to insert: 0
Select operation to perform:
1. Insert
2. In-order traversal
3. Pre-order traversal
4. Exit
1
Enter value to insert: 10
Select operation to perform:
1. Insert
2. In-order traversal
3. Pre-order traversal
4. Exit
1
Enter value to insert: 1
Select operation to perform:
1. Insert
```

```
E:\MODERN\DSA\Practicals\Assignment6\Assign6.exe
Select operation to perform:
1. Insert
2. In-order traversal
3. Pre-order traversal
4. Exit
2
In-order traversal: 0 1 2 4 10
Select operation to perform:
1. Insert
2. In-order traversal
3. Pre-order traversal
4. Exit
3
Pre-order traversal: 4 0 1 2 10
Select operation to perform:
1. Insert
2. In-order traversal
3. Pre-order traversal
4. Exit
4

--------------------------------
Process exited after 91.74 seconds with return value 0
Press any key to continue . . .
```

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