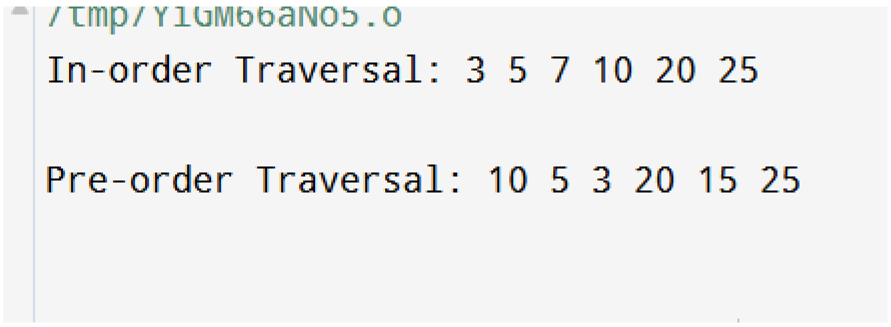
Assignment 6

Implement In-order threaded binary tree and traverse it in In-order and pre-order.

```
#include <iostream>
using namespace std;

// Node structure
struct Node {
    int data;
    Node* left;
    Node* right;
    bool leftThread;
    bool rightThread;
};

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->left = newNode->right = nullptr;
    newNode->leftThread = newNode->rightThread = false;
    return newNode;
}

// Function to insert a node in the threaded binary tree
Node* insert(Node* root, int data) {
    if (root == nullptr) {
        return createNode(data);
    }

    if (data < root->data) {
        root->left = insert(root->left, data);
        if (root->left->right == nullptr) {
            root->left->right = root;
            root->left->rightThread = true;
        }
    } else {
        if (root->rightThread) {
            Node* newNode = createNode(data);
            newNode->right = root->right;
            root->right = newNode;
            root->rightThread = false;
        } else {
            root->right = insert(root->right, data);
        }
    }

    return root;
}

// Function for in-order traversal
void inOrderTraversal(Node*root) {
    Node* current = root;

    // Go to the leftmost node
    while (current && current->left) {
        current = current->left;
    }

    // Traverse the tree
    while (current) {
        cout << current->data << " ";

        // If right is a thread, follow it
        if (!current->leftThread) {
            current = current->right;
        } else {
            // Go to the leftmost node of the right subtree
            current = current->right;
            while (current && current->left) {
                current = current->left;
            }
        }
    }
    cout << endl;
}

// Function for pre-order traversal
void preOrderTraversal(Node* root) {
    if (root == nullptr) {
        return;
    }

    Node* current = root;
    while (current != nullptr) {
        cout << current->data << " ";
        if (!current->leftThread) {
            current = current->left;
        } else if (current->rightThread) {
            break;
        } else {
            current = current->right;
        }
    }

    if (root->right != nullptr && !root->rightThread) {
        preOrderTraversal(root->right);
    }
}

int main() {
    Node* root = nullptr;

    // Inserting nodes into the threaded binary tree
    root = insert(root, 10);
    root = insert(root, 5);
    root = insert(root, 20);
    root = insert(root, 3);
    root = insert(root, 7);
    root = insert(root, 15);
    root = insert(root, 25);

    cout << "In-order Traversal: ";
    inOrderTraversal(root);
    cout << endl;

    cout << "Pre-order Traversal: ";
    preOrderTraversal(root);
    cout << endl;

    return 0;
}
```



```
/tmp/YlGM66aNo5.o
In-order Traversal: 3 5 7 10 20 25

Pre-order Traversal: 10 5 3 20 15 25
```