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

struct ThreadedNode {
    int data;
    ThreadedNode* left;
    ThreadedNode* right;
    bool leftThread;
    bool rightThread;

    ThreadedNode(int val)
        : data(val), left(nullptr), right(nullptr), leftThread(true),
rightThread(true) {}
};

class ThreadedBinaryTree {
private:
    ThreadedNode* root;

    void insert(ThreadedNode*& node, int val) {
        if (!node) {
            node = new ThreadedNode(val);
        } else if (val < node->data) {
            if (!node->leftThread) {
                insert(node->left, val);
            } else {
                ThreadedNode* temp = new ThreadedNode(val);
                temp->left = node->left;
                temp->right = node;
                node->left = temp;
                node->leftThread = false;
            }
        } else {
            if (!node->rightThread) {
                insert(node->right, val);
            } else {
                ThreadedNode* temp = new ThreadedNode(val);
                temp->right = node->right;
                temp->left = node;
                node->right = temp;
                node->rightThread = false;
            }
        }
    }

    ThreadedNode* leftmost(ThreadedNode* node) {
        while (node && !node->leftThread) {
            node = node->left;
        }
        return node;
    }

    void inOrderHelper(ThreadedNode* node) {
        ThreadedNode* current = leftmost(node);
        while (current) {
            cout << current->data << " ";
            if (current->rightThread) {
                current = current->right;
            } else {
                current = leftmost(current->right);
            }
        }
    }

    void preOrderHelper(ThreadedNode* node) {
        ThreadedNode* current = node;
        while (current) {
            cout << current->data << " ";
            if (!current->leftThread) {
                current = current->left;
            } else if (!current->rightThread) {
                current = current->right;
            } else {
                while (current && current->rightThread) {
                    current = current->right;
                }
                if (current) {
                    current = current->right;
                }
            }
        }
    }

public:
    ThreadedBinaryTree() : root(nullptr) {}

    void insert(int val) {
        if (!root) {
            root = new ThreadedNode(val);
        } else {
            insert(root, val);
        }
    }

    void inOrder() {
        cout << "In-order Traversal: ";
        inOrderHelper(root);
        cout << endl;
    }

    void preOrder() {
        cout << "Pre-order Traversal: ";
        preOrderHelper(root);
        cout << endl;
    }
};

int main() {
    ThreadedBinaryTree tbt;
    int baseElements[] = {5, 3, 7, 2, 4, 6, 8};
    for (int val : baseElements) {
        tbt.insert(val);
    }

    tbt.inOrder();
    tbt.preOrder();

    return 0;
}
```