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

struct Node {
   int data;
   Node *left, *right;
   bool isThreaded = false;
};

Node* newNode(int key) {
   Node* node = new Node;
   node->data = key;
   node->left = node->right = nullptr;
   return node;
}

Node* leftMostNode(Node* root) {
   while (root && root->left)
      root = root->left;
   return root;
}

void traverse(Node* root) {
   if (root == nullptr)
      return;

   Node* curr = leftMostNode(root);
   cout << "Inorder Traversal (Threaded): ";
   while (curr) {
      cout << curr->data << " ";
      if (curr->isThreaded)
```

```
            curr = curr->right;
        else
            curr = leftMostNode(curr->right);
    }
    cout << endl;
}

void populateNext(Node* curr, Node*& prev) {
    if (curr == nullptr)
        return;

    populateNext(curr->left, prev);

    if (prev && prev->right == nullptr) {
        prev->right = curr;
        prev->isThreaded = true;
    }

    prev = curr;

    populateNext(curr->right, prev);
}

void convertToThreaded(Node* root) {
    Node* prev = nullptr;
    populateNext(root, prev);
}

int main() {
    Node* root = newNode(5);
    root->left = newNode(2);
```

```
    root->right = newNode(7);
    root->left->left = newNode(1);
    root->left->right = newNode(4);
    root->right->left = newNode(6);
    root->right->right = newNode(9);
    root->left->right->left = newNode(3);
    root->right->right->left = newNode(8);
    root->right->right->right = newNode(10);

    convertToThreaded(root);
    traverse(root);

    return 0;
}
```

Output:

Inorder Traversal (Threaded): 1 2 3 4 5 6 7 8 9 10