```cpp
//BINARY TREE
#include <iostream>
#include <stack>
#include <queue>

using namespace std;

struct Node {
    int key;
    int items;
    Node* left, * right;

    Node(int k = 0) : key(k), items(0), left(nullptr), right(nullptr) {}
};

//Tim trung vi hieu qua
int getItems(Node* node) {
    if (!node) return 0;
    return node->items;
}

int findKth(Node* root, int k) {
    if (!root) return -1;
    int l = getItems(root->left);
    if (l == k) return root->key;
    if (l > k) return findKth(root->left, k);
    else return findKth(root->right, k - l - 1);
}
//


void preOrderNoRecursion(Node* root) {
    stack<Node*> st;
    if (root) st.push(root);

    while (!st.empty()) {
        Node* node = st.top();
        st.pop();

        cout << node->key << " ";
        if (node->right) {
            st.push(node->right);
        }
        if (node->left) {
            st.push(node->left);
        }
    }
}

```

```cpp
void inOrderNoRecursion(Node* root) {
    stack<Node*> st;

    Node* q = root;
    while (q || !st.empty()) {
        while (q) {
            st.push(q);
            q = q->left;
        }
        q = st.top();
        st.pop();
        cout << q->key << " ";
        q = q->right;
    }
}

void postOrderNoRecursion(Node* root) {
    stack<Node*> st1, st2;
    if (root) st1.push(root);

    while (!st1.empty()) {
        Node* node = st1.top();
        st1.pop();

        st2.push(node);
        if (node->left) st1.push(node->left);
        if (node->right) st1.push(node->right);
    }

    while (!st2.empty()) {
        cout << st2.top()->key << " ";
        st2.pop();
    }
}

void removeNodeNoRecursion(Node*& root, int key) {
    Node* z = root, * prev = nullptr;
    while (z && z->key != key) {
        prev = z;
        if (key < z->key) z = z->left;
        else z = z->right;
    }

    if (!z) return;

    Node* y; // y: node that su bi xoa, prev: node cha cua y
    if (!z->left || !z->right) y = z;
    else {
        y = z->left, prev = z;
```

```
 99          while (y->right) prev = y, y = y->right;
100      }
101
102      z->key = y->key;
103
104      if (!prev) root = nullptr;
105      else if (prev->left == y) prev->left = (y->left ? y->left : y->right);
106      else prev->right = (y->left ? y->left : y->right);
107
108      delete y;
109  }
```