```cpp
#include <iostream>

using namespace std;

struct Node {
    int key;
    Node* left = nullptr;
    Node* right = nullptr;
};

Node* tree = nullptr;

Node* search(int key, Node* tree) {
    if(tree == nullptr || tree->key == key) {
        return tree;
    }
    else if(key < tree->key) {
        return search(key, tree->left);
    }
    else {
        return search(key, tree->right);
    }
}



Node* findMin(Node* tree) {
    if(tree == nullptr || tree->left == nullptr) {
        return tree;
    }
    return findMin(tree->left);
}

Node* findMax(Node* tree) {
    if(tree == nullptr || tree->right == nullptr) {
        return tree;
    }
    return findMax(tree->right);
}


Node* predecessor(Node* tree){
    if(tree == nullptr || tree->left == nullptr) {
        return nullptr;
    }
    return findMax(tree->left);
}



Node* successor(Node* tree) {
    if(tree == nullptr || tree->right == nullptr) {
        return nullptr;
    }
    return findMin(tree->right);
}



void insertKey(int key, Node*& tree) {
    if(tree == nullptr) {
        Node* node = new Node;
        node->key = key;
        tree = node;
    }
    else if(key < tree->key) {
        insertKey(key, tree->left);
    }
    else if(key > tree->key) {
        insertKey(key, tree->right);
    }
}



void deleteKey(int key, Node*& tree){
    if(tree == nullptr) {
        return;
    }
    if (key < tree->key) {
        deleteKey(key, tree->left);
    }
    else if (key > tree->key) {
        deleteKey(key, tree->right);
    }
    else {

        if(tree->left != nullptr && tree->right != nullptr) {
            tree->key = successor(tree)->key;
            deleteKey(tree->key, tree->right);
        }
        else {
            Node* temp = tree;
            tree = (tree->left != nullptr) ? tree->left : tree->right;
            delete temp;
        }
    }
}
```

```cpp
void preorder(Node* tree) {
    if(tree == nullptr) {
        return;
    }
    cout<<" "<<tree->key;
    preorder(tree->left);
    preorder(tree->right);
}


void inorder(Node* tree) {
    if(tree == nullptr) {
        return;
    }
    inorder(tree->left);
    cout<<" "<<tree->key;
    inorder(tree->right);
}


void postorder(Node* tree) {
    if(tree == nullptr) {
        return;
    }
    postorder(tree->left);
    postorder(tree->right);
    cout<<" "<<tree->key;
}


int main() {
    int numberOfKeys = 11;
    int keys[numberOfKeys] = {15, 6, 18, 3, 7, 17, 20, 2, 4, 13, 9};
    for(int i = 0; i<numberOfKeys; i++) {
        insertKey(keys[i], tree);
    }

    cout<<"preorder: ";
    preorder(tree);
    cout<<endl;

    cout<<"inorder: ";
    inorder(tree);
    cout<<endl;

    cout<<"postorder: ";
    postorder(tree);
    cout<<endl;

    int searchedKey = 7;
    Node* result = search(searchedKey, tree);
    if(result != nullptr) {
        cout<<"result->key: "<<result->key<<endl;
    }
    else {
        cout<<"searched key "<<searchedKey<<" not found"<<endl;
    }

    searchedKey = 14;
    result = search(searchedKey, tree);
    if(result != nullptr) {
        cout<<"result->key: "<<result->key<<endl;
    }
    else {
        cout<<"searched key "<<searchedKey<<" not found"<<endl;
    }


    cout<<"minimum: "<<findMin(tree)->key<<endl;
    cout<<"maximum: "<<findMax(tree)->key<<endl;
    cout<<"predecessor: "<<predecessor(tree)->key<<endl;
    cout<<"successor: "<<successor(tree)->key<<endl;

    int delKey = 7;
    deleteKey(delKey, tree);
    result = search(delKey, tree);
    if(result != nullptr) {
        cout<<"result->key: "<<result->key<<endl;
    }
    else {
        cout<<"deleted key "<<delKey<<" not found"<<endl;
    }

    cout<<"inorder: ";
    inorder(tree);
    cout<<endl;

    delKey = 9;
    deleteKey(delKey, tree);
    result = search(delKey, tree);
    if(result != nullptr) {
        cout<<"result->key: "<<result->key<<endl;
    }
    else {
        cout<<"deleted key "<<delKey<<" not found"<<endl;
    }

    cout<<"inorder: ";
    inorder(tree);
    cout<<endl;

    delKey = 6;
    deleteKey(delKey, tree);
    result = search(delKey, tree);
    if(result != nullptr) {
        cout<<"result->key: "<<result->key<<endl;
```

```
    }
    else {
        cout<<"deleted key "<<delKey<<" not found"<<endl;
    }

    cout<<"inorder: ";
    inorder(tree);
    cout<<endl;

    while(tree != nullptr) {
        deleteKey(tree->key, tree);
        cout<<"inorder: ";
        inorder(tree);
        cout<<endl;
    }

    return 0;
}
```