

```

1  #include <iostream>
2
3
4  using namespace std;
5
6
7  struct Node {
8      int key;
9      Node* left = nullptr;
10     Node* right = nullptr;
11 };
12
13
14 Node* tree = nullptr;
15
16
17 Node* search(int key, Node* tree) {
18     if(tree == nullptr || tree->key == key) {
19         return tree;
20     }
21     else if(key < tree->key) {
22         return search(key, tree->left);
23     }
24     else {
25         return search(key, tree->right);
26     }
27 }
28
29
30
31 Node* findMin(Node* tree) {
32     if(tree == nullptr || tree->left == nullptr) {
33         return tree;
34     }
35     return findMin(tree->left);
36 }
37
38
39 Node* findMax(Node* tree) {
40     if(tree == nullptr || tree->right == nullptr) {
41         return tree;
42     }
43     return findMax(tree->right);
44 }
45
46
47 Node* predecessor(Node* tree){
48     if(tree == nullptr || tree->left == nullptr) {
49         return nullptr;
50     }
51     return findMax(tree->left);
52 }
53
54
55
56 Node* successor(Node* tree) {
57     if(tree == nullptr || tree->right == nullptr) {
58         return nullptr;
59     }
60     return findMin(tree->right);
61 }
62
63
64
65 void insertKey(int key, Node*& tree) {
66     if(tree == nullptr) {
67         Node* node = new Node;
68         node->key = key;
69         tree = node;
70     }
71     else if(key < tree->key) {
72         insertKey(key, tree->left);
73     }
74     else if(key > tree->key) {
75         insertKey(key, tree->right);
76     }
77 }
78
79
80
81 void deleteKey(int key, Node*& tree){
82     if(tree == nullptr) {
83         return;
84     }
85     if (key < tree->key) {
86         deleteKey(key, tree->left);
87     }
88     else if (key > tree->key) {
89         deleteKey(key, tree->right);
90     }
91     else {
92
93         if(tree->left != nullptr && tree->right != nullptr) {
94             tree->key = successor(tree->key);
95             deleteKey(tree->key, tree->right);
96         }
97         else {
98             Node* temp = tree;
99             tree = (tree->left != nullptr) ? tree->left : tree->right;
100            delete temp;
101        }
102    }
103 }
104
105
106

```

```

107 void preorder(Node* tree) {
108     if(tree == nullptr) {
109         return;
110     }
111     cout<<" "<<tree->key;
112     preorder(tree->left);
113     preorder(tree->right);
114 }
115
116
117 void inorder(Node* tree) {
118     if(tree == nullptr) {
119         return;
120     }
121     inorder(tree->left);
122     cout<<" "<<tree->key;
123     inorder(tree->right);
124 }
125
126
127 void postorder(Node* tree) {
128     if(tree == nullptr) {
129         return;
130     }
131     postorder(tree->left);
132     postorder(tree->right);
133     cout<<" "<<tree->key;
134 }
135
136
137 int main() {
138     int numberOfKeys = 11;
139     int keys[numberOfKeys] = {15, 6, 18, 3, 7, 17, 20, 2, 4, 13, 9};
140     for(int i = 0; i<numberOfKeys; i++) {
141         insertKey(keys[i], tree);
142     }
143
144     cout<<"preorder: ";
145     preorder(tree);
146     cout<<endl;
147
148     cout<<"inorder: ";
149     inorder(tree);
150     cout<<endl;
151
152     cout<<"postorder: ";
153     postorder(tree);
154     cout<<endl;
155
156     int searchedKey = 7;
157     Node* result = search(searchedKey, tree);
158     if(result != nullptr) {
159         cout<<"result->key: "<<result->key<<endl;
160     }
161     else {
162         cout<<"searched key "<<searchedKey<<" not found"<<endl;
163     }
164
165     searchedKey = 14;
166     result = search(searchedKey, tree);
167     if(result != nullptr) {
168         cout<<"result->key: "<<result->key<<endl;
169     }
170     else {
171         cout<<"searched key "<<searchedKey<<" not found"<<endl;
172     }
173
174
175     cout<<"minimum: "<<findMin(tree)->key<<endl;
176     cout<<"maximum: "<<findMax(tree)->key<<endl;
177     cout<<"predecessor: "<<predecessor(tree)->key<<endl;
178     cout<<"successor: "<<successor(tree)->key<<endl;
179
180     int delKey = 7;
181     deleteKey(delKey, tree);
182     result = search(delKey, tree);
183     if(result != nullptr) {
184         cout<<"result->key: "<<result->key<<endl;
185     }
186     else {
187         cout<<"deleted key "<<delKey<<" not found"<<endl;
188     }
189
190     cout<<"inorder: ";
191     inorder(tree);
192     cout<<endl;
193
194     delKey = 9;
195     deleteKey(delKey, tree);
196     result = search(delKey, tree);
197     if(result != nullptr) {
198         cout<<"result->key: "<<result->key<<endl;
199     }
200     else {
201         cout<<"deleted key "<<delKey<<" not found"<<endl;
202     }
203
204     cout<<"inorder: ";
205     inorder(tree);
206     cout<<endl;
207
208     delKey = 6;
209     deleteKey(delKey, tree);
210     result = search(delKey, tree);
211     if(result != nullptr) {
212         cout<<"result->key: "<<result->key<<endl;

```

```
213     }
214     else {
215         cout<<"deleted key "<<delKey<<" not found"<<endl;
216     }
217
218     cout<<"inorder: ";
219     inorder(tree);
220     cout<<endl;
221
222     while(tree != nullptr) {
223         deleteKey(tree->key, tree);
224         cout<<"inorder: ";
225         inorder(tree);
226         cout<<endl;
227     }
228
229     return 0;
230 }
```