```cpp
#include<iostream>
#include<iomanip>
using namespace std;
typedef struct {
    int key;
}ElemType;
typedef struct BiTNode {
    ElemType data;
    BiTNode *lchild, *rchild;
    int depth;
}BiTNode, *BiTree;
double n;
BiTree T;
double sum = 0.0;
void InsertBST(BiTree &T, BiTree S)
{
    BiTree p, q = NULL;
    if (!T) {
        T = S;
    }
    else
    {
        p = T;
        while (p)
        {
            q = p;
            if (p->data.key > S->data.key)
                p = p->lchild;
            else
                p = p->rchild;
        }
        if (q->data.key > S->data.key)
            q->lchild = S;
        else
            q->rchild = S;
    }
}
void CreateBST(BiTree &T)
{
    BiTree S;
    int x, num = 0;
    while (num < n)
    {
        cin >> x;
```

```
            S = new BiTNode;
            S->data.key = x;
            S->lchild = NULL;
            S->rchild = NULL;
            InsertBST(T, S);
            num++;
        }
}
int SearchBST(BiTree &T, int key)
{
    BiTree p = NULL;
    if (!T)
    {
        p = T;
        while (p)
        {
            if (p->data.key == key)
                break;
            else if (p->data.key > key)
                p = p->lchild;
            else
                p = p->rchild;
        }
    }
    if (p == NULL)
        return 0;
    else
        return 1;
}
int DeleteNode(BiTree &p)
{
    BiTree s, q;
    int flag = 0;
    if (!p->lchild)
    {
        q = p;
        p = p->rchild;
        flag = 1;
        delete q;
    }
    else if (!p->rchild)
    {
        q = p;
        p = p->lchild;
        flag = 1;
        delete q;
    }
    else
    {
        q = p;
        s = p->lchild;
        while (s->rchild)
        {
            q = s;
            s = s->rchild;
        }
        p->data = s->data;
        if (p != q)
            q->rchild = s->lchild;
        else
            q->lchild = s->lchild;
        flag = 1;
        delete s;
    }
    if (flag == 1)
        return 1;
    else
        return 0;
}
int DeleteBST(BiTree &T, int key) {
    if (!T) return 0;
    else {
        if (key == T->data.key) { return DeleteNode(T); }
        else if (key < T->data.key) return DeleteBST(T->lchild, key);
        else return DeleteBST(T->rchild, key);
    }
}
void PreOrderTraverse(BiTree &T)
{
    if (!T)
        return;
    else
    {
        cout << T->data.key << " ";
        PreOrderTraverse(T->lchild);
        PreOrderTraverse(T->rchild);
    }
}
```

```
void NodeDepth(BiTree &T, int Depth)
{
    if (T)
    {
        NodeDepth(T->rchild, Depth + 1);
        T->depth = Depth;
        NodeDepth(T->lchild, Depth + 1);
    }
}
int SumSearchLength(BiTree &T)
{
    if (!T)
        return sum;
    else
    {
        sum += T->depth;
        SumSearchLength(T->lchild);
        SumSearchLength(T->rchild);
    }
}
int main()
{
    int key;
    cin >> n;
    CreateBST(T);
    cin >> key;
    cout << DeleteBST(T, key);
    cout << endl;
    cout << SearchBST(T, key);
    cout << endl;
    BiTree S;
    S = new BiTNode;
    S->data.key = key;
    S->lchild = NULL;
    S->rchild = NULL;
    InsertBST(T, S);
    PreOrderTraverse(T);
    cout << endl;
    double Depth = 1;
    NodeDepth(T, Depth);
    cout << setiosflags(ios::fixed) << setprecision(2) <<
SumSearchLength(T) / n << endl;
    return 0;
}
```