```c
#include<stdio.h>
#include<stdlib.h>

struct  node* t=NULL;

struct node{
    int data;
    struct node* left;
    struct node* right;
    struct node* parent;

};

struct node* cnode(int v){
    struct node* temp=(struct node*)malloc(sizeof(struct node));
    temp->data=v;
    temp->left=temp->right=NULL;
    temp->parent=NULL;
    return(temp);
}

/*int maxof(int a,int b){
    if(a>b)return a;
    return b;
}

void rotate_right(struct node* t){
    int x=t->data;
    int y=t->left->data;
    struct node* tll=t->left->left;
    struct node* tlr=t->left->right;
    struct node* tr=t->right;

    t->data=y;
    t->right=t->left;
    t->right->data=x;
    t->left=tll;
    t->right->left=tlr;
    t->right->right=tr;
}

void rotate_left(struct node* t){
    int y=t->data;
    int z=t->right->data;
    struct node* tll=t->left;
    struct node* tlrl=t->right->left;
    struct node* tlrr=t->right->right;

    t->data=z;
    t->left=t->right;
    t->left->data=y;
    t->left->left=tll;
    t->left->right=tlrl;
    t->right=tlrr;
```

```
}

int slope(struct node* t){
    return(t->left->ht-t->right->ht);
}

void rebalance(struct node* t){
    if(slope(t)==2){
        if(slope(t->left)==-1)
          rotate_left(t->left);
    rotate_right(t);
     }
    if(slope(t)==-2){
         if(slope(t->right)==1)
            rotate_right(t->right);
    rotate_left(t);
    }
    return;
}*/
struct node* min(struct node* t){
    struct node* temp=t;
    while(temp->left!=NULL){
        temp=temp->left;
    }
    return(temp);
}
struct node* max(struct node* t){
    struct node* temp=t;
    while(temp->right!=NULL){
        temp=temp->right;
    }
    return(temp);
}
struct node* find(struct node* t,int v){
    if(v==t->data)
      return t;
    else if(v<t->data){
        find(t->left,v);
    }
    else{
        find(t->right,v);
    }
}
struct node* succ(struct node* t,int v){
    struct node* temp=find(t,v);
    if(temp->right!=NULL){
        return(min(temp->right));
    }
    struct node* temp2=temp->parent;
    while(temp2->parent!=NULL&&temp2->right==temp){
        temp=temp2;
        temp2=temp2->parent;

    }
```

```c
    if(temp2->parent==NULL&&temp2->right==temp){
        return temp2->parent;
    }
    return(temp2);


}
struct node* pred(struct node* t,int v){
    struct node* temp=find(t,v);
    if(temp->left!=NULL){
        return(max(temp->left));
    }
    struct node* temp2=temp->parent;
    while(temp2->parent!=NULL&&temp2->left==temp){
        temp=temp2;
        temp2=temp2->parent;
    }
    if(temp2->parent==NULL&&temp2->left==temp){
        return(temp2->parent);
    }
    return(temp2);
}

struct node* insert(struct node* t,int v){
    if(t==NULL){
        t=cnode(v);
        return t;
    }
    else if(v<t->data){

        t->left=insert(t->left,v);
        //rebalance(t->left);
    //  t->ht=1+maxof(t->left->ht,t->right->ht);
        t->left->parent=t;
    }
    else if(v>t->data){
        t->right=insert(t->right,v);
        //rebalance(t->right);
    //  t->ht=1+maxof(t->left->ht,t->right->ht);
        t->right->parent=t;
    }
    return t;
}

struct node* del(struct node* t,int v){
    if(t==NULL)
       return t;
    if(v<t->data){
        t->left=del(t->left,v);
    }
    else if(v>t->data){
        t->right=del(t->right,v);
    }
    else{
        if(t->left==NULL){
            struct node* temp=t->right;
            free(t);
            return temp;
        }
        else if(t->right==NULL){
            struct node* temp=t->left;
            free(t);
            return temp;
        }
        else if(t->left!=NULL&&t->right!=NULL){
             struct node* p=min(t);
            t->data=p->data;
            t->right=del(t->right,t->data);
        }

    }
}
void inorder(struct node* t){
    if(t!=NULL){
        inorder(t->left);
        printf("%d::\n",t->data);
        inorder(t->right);
    }
}
int main(){
    t=insert(t,5);
    t=insert(t,3);
    t=insert(t,10);
    t=insert(t,1);
    t=insert(t,2);
    t=insert(t,4);
    t=insert(t,11);
    inorder(t);
   int p;
    printf("\n");
    printf("For which value you want to know SUccessor: ");
    scanf("%d",&p);
    struct node* k=succ(t,p);

    if(k!=NULL){

        printf(" SUCCESSOR od %d IS: %d \n",p,k->data);
    }
    else{
        printf("NO SUCCESSOR EXIST for %d!",p);
    }
    k=NULL;
    printf("\n");
    printf("For which value you want to know predeccor: ");
    scanf("%d",&p);
    k=pred(t,p);
    if(k!=NULL){

        printf("\npredeccsor of %d IS: %d \n",p,k->data);
    }
    else{
        printf("\nNO predesccor EXIST!");
    }

    printf("\n");
}
```