Vishwas Gajwada, 197188

**E1\_35**

Code:

#include<iostream>

using namespace std;

#define null NULL

typedef struct node \* bptr;

struct node{

    int l=0,r=0;

    bptr lc=null,rc=null;

};

void addRange(bptr &t,int left,int right)

{

    if(!t){

        t=new node;

        t->l=left;

        t->r=right;

        return;

    }

    if(t->l<=left && right<=t->r)return;

    if(left<t->l && right<=t->r){

        if(right<t->l)addRange(t->lc,left,right);

        else addRange(t->lc,left,t->l);return;

    }

    if(t->l<=left && t->r<right){

        if(left>t->r)addRange(t->rc,left,right);

        else addRange(t->rc,t->r,right);return;

    }

    if(left<t->l && t->r<right){

        addRange(t->lc,left,t->l);

        addRange(t->rc,t->r,right);return;

    }

}

bool searchRange(bptr t,int left,int right){

    if(!t)return false;

    if(t->l<=left && right<=t->r)return true;

    if(left<t->l && right<=t->r){

        if(right<t->l)return searchRange(t->lc,left,right);

        return searchRange(t->lc,left,t->l);

    }

    if(t->l<=left && t->r<right){

        if(left>t->r)return searchRange(t->rc,left,right);

        return searchRange(t->rc,t->r,right);

    }

    if(left<t->l && t->r<right){

        return searchRange(t->lc,left,t->l) && searchRange(t->rc,t->r,right);

    }

    return false;

}

void deleteRange(bptr &t,int left,int right);

void delfull(bptr &t){

    if(!t->lc){t=t->rc;return;}

    else if(!t->rc){t=t->lc;return;}

    else{

        bptr lmax=t->lc;

        while(lmax->rc)lmax=lmax->rc;

        t->l=lmax->l;

        t->r=lmax->r;

        deleteRange(t->lc,lmax->l,lmax->r);

    }

}

void deleteRange(bptr &t,int left,int right){

    if(!t)return;

    if(t->l<=left && right<=t->r){

        if(t->l==left && t->r==right){

            delfull(t);return;

        }

        if(t->l==left){t->l=right;return;}

        if(t->r==right){t->r=left;return;}

        bptr temp = new node;

        temp->l=right;temp->r=t->r;

        temp->rc=t->rc;

        t->rc=temp;

        t->r=left;

        return;

    }

    if(right<t->l){deleteRange(t->lc,left,right);return;}

    if(left>t->r){deleteRange(t->rc,left,right);return;}

    if(left<t->l && right<t->r){

        deleteRange(t,t->l,right);

        deleteRange(t->lc,left,t->l);return;

    }

    if(left>t->l && right>t->r){

        deleteRange(t,left,t->r);

        deleteRange(t->lc,t->r,right);return;

    }

    if(left<t->l && right>t->r){

        deleteRange(t,t->l,t->r);

        deleteRange(t->lc,left,t->l);

        deleteRange(t->rc,t->r,right);return;

    }

}

int main()

{

    bptr t=null;

    addRange(t,10,20);

    deleteRange(t,14,16);

    if(searchRange(t,10,14))cout<<"true";else cout<<"false";cout<<endl;

    if(searchRange(t,13,15))cout<<"true";else cout<<"false";cout<<endl;

    if(searchRange(t,16,17))cout<<"true";else cout<<"false";cout<<endl;

}

Output:

