Vishwas Gajawada , 197188

B-3

Code:

#include<bits/stdc++.h>

using namespace std;

#define null NULL

typedef struct Bdnode \* bdptr;

struct Bdnode

{

   int cnt=0;

   int \*keys;

   bdptr \*cptr;

   bool leaf=true;

};

void split(bdptr &l,bdptr &r,int &mid,int d,int &bias)

{

    int mid\_ind;

    if(bias==0){//right biased

        mid\_ind=d/2-1;

        bias=1;

    }else if(bias==1){//left biased

        mid\_ind=d/2;

        bias=0;

    }

    if(d%2!=0)mid\_ind = (d)/2;

    r->leaf=l->leaf;

    for(int i=mid\_ind+1;i<d;i++)

    {

        r->cptr[r->cnt]=l->cptr[i];

        r->keys[r->cnt++]=l->keys[i];

        l->cptr[i]=null;

        l->keys[i]=0;

        l->cnt--;

    }

    r->cptr[r->cnt]=l->cptr[d];

    l->cptr[d]=null;

    mid=l->keys[mid\_ind];

    l->keys[mid\_ind]=0;

    l->cnt--;

}

void insert(bdptr &bd,int x,int d)

{

    int i=bd->cnt-1;

    while(i>=0 && bd->keys[i]>x)

    {

        bd->keys[i+1]=bd->keys[i];

        bd->cptr[i+2]=bd->cptr[i+1];

        i--;

    }

    bd->keys[i+1]=x;

    bd->cptr[i+2]=bd->cptr[i+1];

    bd->cnt++;

}

bdptr kothadi(int d)

{

    bdptr bd = new Bdnode;

    bd->keys=new int[d];// one extra

    bd->cptr = new bdptr[d+1];//one extra

    for(int i=0;i<d+1;i++)bd->cptr[i]=null;

    return bd;

}

void create(bdptr &bd,int &x,int d,bdptr parent,bdptr &head,int &upOrdown,bdptr &left,bdptr &right,int &bias)

{

    if(!bd)

    {

        bd = kothadi(d);

        bd->keys[bd->cnt++]=x;

        return;

    }

    int ind=-1;

    for(int i=bd->cnt;i>=0;i--)

    {

        if(i==0 && x<bd->keys[0])ind=0;

        if(i!=0 && x > bd->keys[i-1]){ind=i;break;}

    }

    if(bd->cptr[ind]) create(bd->cptr[ind],x,d,bd,head,upOrdown,left,right,bias);//go down till leaf

    else insert(bd,x,d);//is leaf

    if(upOrdown==1){ //coming back from recursion and wants to add mid of child's overflow to current

        insert(bd,x,d);

        ind=-1;

        for(int i=0;i<bd->cnt;i++)

        {

            if(bd->keys[i]==x)ind=i;

        }

        bd->cptr[ind]=left;

        bd->cptr[ind+1]=right;

        upOrdown=0;

    }

    if(bd->cnt==d)//overflow

    {

        bdptr r=kothadi(d);

        split(bd,r,x,d,bias);

        left=bd;right=r;

        if(parent==null)

        {

            bdptr par = kothadi(d);

            par->keys[par->cnt++]=x;

            par->cptr[0]=left;

            par->cptr[1]=right;

            par->leaf=false;

            head=par;

        }

        upOrdown=1;

        return;

    }

}

void create(bdptr &bd,int x,int d,int &bias)

{

    int upordown=0;

    bdptr left=null,right=null;

    create(bd,x,d,null,bd,upordown,left,right,bias);

}

void levelorder(bdptr bd,int d)

{

    queue<bdptr>q;

    bdptr end = kothadi(d);

    end->keys[end->cnt++]=-1;

    q.push(bd);q.push(end);

    while(true)

    {

        bdptr temp = q.front();q.pop();

        if(temp->keys[0]==-1)

        {

            if(q.empty())break;

            cout<<endl;q.push(end);

        }else{

            for(int i=0;i<temp->cnt;i++)cout<<temp->keys[i]<<" ";

            for(int i=0;i<=temp->cnt;i++)

                if(temp->cptr[i])q.push(temp->cptr[i]);

        }

    }

}

int main()

{

    bdptr BD=null;

    int d,n;

    cin>>d;

    int bias=1;//left

    cin>>n;

    while(n!=-1)

    {

        create(BD,n,d,bias);

        cin>>n;

    }

    levelorder(BD,d);

}

// 4 12 11 10 9 8 7 6 5 4 3 2 1 -1

Input:

4

12 11 10 9 8 7 6 5 4 3 2 1 -1

Output:

