C1-3-197188

```
#include<bits/stdc++.h>
using namespace std;
#define null NULL
typedef struct Bdnode * bdptr;
typedef struct lnode * lptr;
struct lnode{
    int data;
    lptr next=null;
};
struct Bdnode
{
   int cnt=0;
   lptr *keys;
   bdptr *cptr;
   bool leaf=true;
};
int last(lptr l)
{
    if(!l)return 0;
    while(l->next)l=l->next;
    return l->data;
}

void split(bdptr &l,bdptr &r,lptr &mid,int d)
{
    int mid_ind = d/2-1;//left bias,for right bias take d/2
    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]=null;
        l->cnt--;
    }
    r->cptr[r->cnt]=l->cptr[d];
    l->cptr[d]=null;
    mid=l->keys[mid_ind];
    l->keys[mid_ind]=null;
    l->cnt--;
}
void addend(lptr &l,int x)
{
    lptr temp = new(lnode);
    temp->data=x;
    if(!l){
        l=temp;
        return;
    }
    lptr tail=l;
    while(tail->next)tail=tail->next;
    tail->next=temp;
}
```

```
void insert(bdptr &bd,lptr x,int d)
{
    int i=bd->cnt-1;
    while(i>=0 && last(bd->keys[i])>last(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 lptr[d];// one extra
    bd->cptr = new bdptr[d+1];//one extra
    for(int i=0;i<d;i++)bd->keys[i]=null;
    for(int i=0;i<d+1;i++)bd->cptr[i]=null;
    return bd;
}
void create(bdptr &bd,lptr &x,int d,bdptr parent,bdptr &head,int
&upOrdown,bdptr &left,bdptr &right)
{
    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 && last(x)<last(bd->keys[0]))ind=0;
        if(i!=0 && last(x) > last(bd->keys[i-1])){ind=i;break;}
    }
    if(bd->cptr[ind]) create(bd-
>cptr[ind],x,d,bd,head,upOrdown,left,right);//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);
            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,lptr x,int d)
    {
        int upordown=0;
        bdptr left=null,right=null;
        create(bd,x,d,null,bd,upordown,left,right);
    }
    void print(lptr l)
    {
        if(!l)return;
        while(l)
        {
            cout<<l->data<<" ";
            l=l->next;
        }
        cout<<endl;
    }
    void levelorder(bdptr bd,int d)
    {
        queue<bdptr>q;
        bdptr end = kothadi(d);
        addend(end->keys[end->cnt++],-1);
        q.push(bd);q.push(end);
        while(true)
        {
            bdptr temp = q.front();q.pop();
            if(temp->keys[0]->data==-1)
            {
                if(q.empty())break;
                q.push(end);
            }else{
                for(int i=0;i<temp->cnt;i++)print(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=3;
    for(int i=0;i<10;i++)
    {
        lptr l=null;
        int n;cin>>n;
        while(n!=-1){
            addend(l,n);cin>>n;
        }
        create(BD,l,d);
    }
    levelorder(BD,d);
}
```

INPUT:

```
1 2 3 4 5 6 7 8 9 10 -1
1 2 3 4 5 6 7 8 9 -1
1 2 3 4 5 6 7 8 -1
1 2 3 4 5 6 7 -1
1 2 3 4 5 6 -1
1 2 3 4 5 -1
1 2 3 4 -1
1 2 3 -1
1 2 -1
1 -1
```

OUTPUT:

```
PS C:\Users\Vishwas Gajawada\Desktop\c++ codes\dsa midlab> cd "c:\User
-3 }
1 2 3 4 5 6 7 8 9 10 -1
1 2 3 4 5 6 7 8 9 -1
1 2 3 4 5 6 7 8 -1
1 2 3 4 5 6 7 -1
1 2 3 4 5 6 -1
1 2 3 4 5 -1
1 2 3 4 -1
1 2 3 -1
1 2 -1
1 -1
1 2 3 4 5 6 7
1 2 3
1 2 3 4 5
1 2 3 4 5 6 7 8 9
1
1 2
1 2 3 4
1 2 3 4 5 6
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9 10
PS C:\Users\Vishwas Gajawada\Desktop\c++ codes\dsa midlab>
```