**Complete binary tree and deletion :**

#include <iostream>

#include <cmath>

using namespace std;

struct node

{

int info;

node \*left,\*right;

};

class Btree\_insertion

{

private :

node \*temp,\*temp2;

int count;

public :

int number;

node \*root;

int h,c\_h,y,i,flag,g\_h;

Btree\_insertion()

{

temp=NULL;

temp2=NULL;

root=NULL;

c\_h=0;

g\_h=0;

h=0;

count=0;

}

int insertion(node \*temp)

{

i=0;

if(root==NULL)

{

cout<<"Enter in root...";

root=new node;

root->info=number;

root->left=NULL;

root->right=NULL;

count++;

i=1;

return 1;

}

if(temp->left!=NULL )

{

c\_h++;

insertion(temp->left);

c\_h--;

}

y=h+1;

if(temp->left==NULL && count==(pow(2.0,y)-1))

{

cout<<"info enter in left and hight is "<<h<<" : ";

temp2=new node;

temp2->info=number;

temp2->left=NULL;

temp2->right=NULL;

temp->left=temp2;

count++;

h++;

i=1;

return 1;

}

if(temp->left!=NULL && temp->right==NULL)

{

cout<<"inf enter in right...";

temp2=new node;

temp2->info=number;

temp2->left=NULL;

temp2->right=NULL;

temp->right=temp2;

count++;

i=1;

return 1;

}

if(c\_h < h &&temp->left==NULL)

{

cout<<"info enter in right subtree in left:";

temp2=new node;

temp2->info=number;

temp2->left=NULL;

temp2->right=NULL;

temp->left=temp2;

count++;

i=1;

return 1;

}

if(temp->right!=NULL && i==0)

{

c\_h++;

insertion(temp->right);

c\_h--;

}

}

int deletion(node \*temp)

{

flag=0;

if(root==NULL)

{

cout<<"tree is empty...";

return 1;

}

if(root->left==NULL && root->right==NULL)

{

cout<<"root deleted...";

delete root;

count--;

root=NULL;

flag=1;

return 1;

}

if(temp->right!=NULL && flag==0 )

{

g\_h++;

deletion(temp->right);

g\_h--;

}

if(count==pow(2.0,h) && temp->left!=NULL && temp->right==NULL)

{

cout<<"left info deleted and hight is decreases...";

temp2=temp->left;

delete temp2;

temp->left=NULL;

h--;

count--;

flag=1;

return 1;

}

if(h-g\_h==1 && temp->right!=NULL &&temp->left->left==NULL)

{

cout<<"right node deleted...";

temp2=temp->right;

delete temp2;

temp->right=NULL;

count--;

flag=1;

return 1;

}

if(temp->left!=NULL &&temp->right==NULL)

{

cout<<"left node deleted...";

temp2=temp->left;

delete temp2;

temp->left=NULL;

count--;

flag=1;

return 1;

}

if(temp->left!=NULL && flag==0)

{

g\_h++;

deletion(temp->left);

g\_h--;

}

return 1;

}

void print\_inorder(node \*temp)

{

if(root==NULL)

{

cout<<"Tree is empty :";

return;

}

if(temp->left!=NULL)

print\_inorder(temp->left);

cout<<temp->info<<" ";

if(temp->right!=NULL)

print\_inorder(temp->right);

return;

}

};

int main()

{

Btree\_insertion obj;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";

cout<<"\nEnter 1 for insertion\n";

cout<<"Enter 2 fo print \n";

cout<<"Enter 3 for deletion\n";

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";

int num=0;

while(num!=4)

{

cout<<endl;

cin>>num;

if(num==1)

{

cout<<"Enter number :";

cin>>obj.number;

obj.insertion(obj.root);

}

if(num==2)

{

cout<<"tree print in inorder :";

obj.print\_inorder(obj.root);

}

if(num==3)

{

cout<<"deletetion function call... ";

obj.deletion(obj.root);

}

}

}