BINARY_TREE.cpp

```
/*ROLL NUMBER : 2002
BATCH : E-10*/

#include <iostream>
#include<stdlib.h>
#include "b_tree.h"
using namespace std;

int main()
{
        b_tree b;
        node *r;

        node *p;
        node *copy;
        do
                {
                        int ch;
                        cout<<"\t----------------MENU------------------\n";

                        cout<<"\t\t1.Create\n\t\t2.Depth of Tree\n\t\t3.Display Leaves\n\t\t4.Display
tree\n\t\t"
                                        "5.Create a copy\n\t\t6.Level Display\n\t\t7.Exit\n\n";
                        cout<<"Enter Choice : ";
                        cin>>ch;
                        cout<<endl;

                        switch(ch)
                        {
                                case 1: r=b.create();
                                        break;

                                case 2: cout<<"Height of tree is : "<<b.height(r)<<endl;
                                                break;

                                case 3: cout<<"Leaf Nodes Are : ";
                                                p=b.display_leaf(r);
                                                cout<<"\n\n";
                                                break;

                                case 4: cout<<"Tree in preorder is :: \n";
                                                b.display(r);
                                                cout<<"\n\n";
                                                break;

                                case 5: copy=b.copy_tree(r);
                                                cout<<"\t\t!! Copied !!\n\n";
                                                b.display(copy);
```

```
                                        break;

                                case 6: cout<<"Tree in level order is :: \n";
                                        b.display_level(r);
                                        cout<<"\n\n";
                                        break;

                                case 7: exit(0);
                                        break;

                        }
                }
                while(1);

        return 0;
}
```

B_TREE.h

```
/*
 * b_tree.h
 *
 *  Created on: 15-Jan-2018
 *      Author: e2002
 */

#ifndef B_TREE_H_
#define B_TREE_H_

struct node
{
        node *left;
        node *right;

        int data;
};

class b_tree
{
        node * root;
        public:
                        node* create();
                        node* display_leaf(node *);
                        int height(node *);
                        void display(node *);
                        void display_level(node *);
                        node *copy_tree(node *);
};

#endif /* B_TREE_H_ */
```

B_TREE.cpp

```
/*
 * b_tree.cpp
 *
 *  Created on: 15-Jan-2018
 *      Author: e2002
 */

#include "b_tree.h"
#include<bits/stdc++.h>
#include<iostream>
#include<malloc.h>
#include<queue>
using namespace std;

node * b_tree :: create()
{
	node *p;
	int x;

	cout<<"Enter Element (-1 for null data) of node : ";
	cin>>x;

	if(x==-1)
		return NULL;

	p=new node;

	p->data=x;

	cout<<"Left node of "<<x<<" :: \n\t";
	p->left=create();
	cout<<"Right node of "<<x<<" :: \n\t";
	p->right=create();

	return p;

}


node * b_tree :: display_leaf(node * root)
{
	node *temp;
	temp=root;

	if(temp != NULL)
	{
		if(temp->right==NULL && temp->left==NULL)
			cout<<""<<temp->data<<" ";

		else
		{
			display_leaf(temp->right);
```

```
                        display_leaf(temp->left);
                }
        }
        return temp;
}

int b_tree :: height(node * root)
{
        if(root==NULL)
                return 0;

        int left_height=0,right_height=0;

        left_height =height(root->left)+1;
        right_height=height(root->right)+1;

        return (left_height>right_height) ?  left_height :  right_height;
}

node * b_tree :: copy_tree(node * root)
{
        if(root==NULL)
                        return NULL;
        node * copy;

        copy=new node;

        copy->data=root->data;

        copy->left=copy_tree(root->left);
        copy->right=copy_tree(root->right);

        return copy;
}

void b_tree :: display(node * root)
{
        if(root==NULL)
                return ;

        cout<<root->data<<" ";

        display(root->left);
        display(root->right);
}

void b_tree :: display_level(node * root)
{
        queue<node *> q1, q2;

    if (root == NULL)
       return;
```

```
    q1.push(root);

    while (!q1.empty() || !q2.empty())
    {
        while (!q1.empty())
        {
            if (q1.front()->left != NULL)
                q2.push(q1.front()->left);

            if (q1.front()->right != NULL)
                q2.push(q1.front()->right);

            cout << q1.front()->data << " ";
            q1.pop();
        }

        cout << "\n";

        while (!q2.empty())
        {
            if (q2.front()->left != NULL)
                q1.push(q2.front()->left);

            if (q2.front()->right != NULL)
                q1.push(q2.front()->right);

            cout << q2.front()->data << " ";
            q2.pop();
        }

        cout << "\n";
    }
}
```