ASSIGNMENT NO. 3

```
#include <iostream>
using namespace std;

struct node
{
	int data;
	node *left,*right;
};


class tree
{
	public:
		node *root,*temp;

		tree()
		{
		 root=NULL;
		}

		int height1(node * root);
		node * mirror1(node *root);

		void create();
		void insert(node * root,node *temp);
		void inorder(node *root);

		void min(node * root);
		node * search(node * root,int x );
};




int tree::height1(node *T)
{
		if(T==NULL)
		return(-1);

		if(T->left==NULL && T->right==NULL)
		return(0);

		return( max(height1(T->left), height1(T->right)) +1);
}



void tree::create()
{
	int n,i;
	root=NULL;
	char ch;
```

```
        cout<<"\nHow many nodes you want in a tree?";
        cin>>n;
        for(i=0;i<n;i++)
        {
        temp=new node;
        cout<<" enter data "<<endl;
        cin>>temp->data;
        temp->left=NULL;
        temp->right=NULL;
        if(root==NULL)
        root=temp;
        else
        {
                insert(root,temp);
        }

        }

}//create end



void tree::insert(node *root,node *temp)
{

        if(temp->data<root->data)
        {

        if(root->left==NULL)
        root->left=temp;
        else
        insert(root->left,temp);
        }

        else if(temp->data>root->data)
        {
                if(root->right==NULL)
                root->right=temp;
                else
                insert(root->right,temp);
        }
}
node * tree::mirror1(node     *T)
{
        node *temp;
        if(T==NULL)
                return NULL;
        else
        {
                temp=T->left;
                T->left=mirror1(T->right);
                T->right=mirror1(temp);
                return T;
        }
}



void tree::inorder(node *root)
{
```

```
        if(root!=NULL)
  {
        inorder(root->left);
        cout<<" "<<root->data;
        inorder(root->right);
  }
}

void tree::min(node * root)
{
        while(root->left!=NULL)
        root=root->left;
        cout<<"Minimum Node in the tree is= "<<root->data;
}

node * tree::search(node * root,int x)
{

        while(root!=NULL)
         {
                if(x<root->data)
                {
                        root=root->left;
                }
                else if(x>root->data)
                {
                        root=root->right;
                }
                else if(x==root->data)
                {
                        break;

                }
        }
        return(root);
}

int main()
{
        tree t1;
        node *temp;
        int xx,op,x,c;
        do
{
        cout<<"\n\n1)Create\n2)Insert\n3)Mirror";
        cout<<"\n4)No of nodes in longest Path\n5)inorder display\n6.minimum
value\n7.Search\n8.Exit";
        cout <<"\nEnter Your Choice :"<<endl;
        cin>>op;
        switch(op)
        {
        case 1:
                        t1.create();
                        break;
        case 2:
                        temp=new node;
                        cout<<" enter data"<<endl;
                        cin>>temp->data;
                        temp->left=NULL;
                        temp->right=NULL;
```

```
                        if(t1.root==NULL)
                        t1.root=temp;
                        else
                        {
                                t1.insert(t1.root,temp);
                        }
                        break;
        case 3:
                        cout<<"\n Original tree in inorder :\n";
                        t1.inorder(t1.root);
                        t1.root=t1.mirror1(t1.root);
                        cout<<"\n Mirrored tree in inorder :\n";
                        t1.inorder(t1.root);

                        cout<<"\nOriginal treerestored";
                        t1.root=t1.mirror1(t1.root);
                        break;
        case 4:
                        cout<<"\n Number of nodes in longest path = "<<t1.height1(t1.root)+1;

                        break;

        case 5:
                        t1.inorder(t1.root);
                        break;
        case 6:

                        t1.min(t1.root);
                        break;
        case 7:
                        cout<<"enter element to search";
                        cin>>x;
                        temp=t1.search(t1.root,x);
                        if(temp==NULL)
                                cout<<"data not found";
                        else
                           cout<<"Data Found";
                        break;
        case 8:
                        exit(0);
        }
}
while(op!=8);
return 0;
}
```

****

/* Output:

```
1)Create
2)Insert
3)Mirror
```

```
4)No of nodes in longest Path
5)inorder display
6.minimum value
7.Search
8.Exit
Enter Your Choice :
1


How many nodes you want in a tree?4
 enter data
12
 enter data
4
 enter data
6
 enter data
25


1)Create
2)Insert
3)Mirror
4)No of nodes in longest Path
5)inorder display
6.minimum value
7.Search
8.Exit

Enter Your Choice :
2
 enter data
20


1)Create
2)Insert
3)Mirror
4)No of nodes in longest Path
5)inorder display
6.minimum value
7.Search
8.Exit

Enter Your Choice :
5
 4 6 12 20 25


1)Create
2)Insert
3)Mirror
4)No of nodes in longest Path
5)inorder display
6.minimum value
7.Search
8.Exit

Enter Your Choice :
3
```

```
 Original tree in inorder :
 4 6 12 20 25
 Mirrored tree in inorder :
 25 20 12 6 4
Original treerestored



1)Create
2)Insert
3)Mirror
4)No of nodes in longest Path
5)inorder display
6.minimum value
7.Search
8.Exit

Enter Your Choice :
4


 Number of nodes in longest path = 3

*/
```