```
//============================================================================
// Name        : BST.cpp
// Author      :
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
//============================================================================

#include <iostream>
using namespace std;

class BTNODE
{
      int data;
      BTNODE *left;
      BTNODE *right;
public:
      BTNODE()
{
            left=right=NULL;data=-1;
}
      friend class BST;
};
      class BST
      {
            BTNODE *root;
            BST()
            {
                  root=NULL;
            }
            void create();
            void inorder(BTNODE *);
            void insert(BTNODE *);
            BTNODE *mirror(BTNODE *T);
            int height(BTNODE *);
            void minvalue(BTNODE *);
            void maxvalue(BTNODE *);

            BTNODE *search(BTNODE *T,int key)
             {
                        while(T!=NULL)
                        {
                              if(T->data==key)
                                    //cout<<T->data;
                                     return T;
                              if(key<T->data)
                                     T=T->left;
                              else
                                    T=T->right;
                        }
                        return NULL;
                 }

            friend int main();
      };
      void BST:: minvalue(BTNODE *T)
```

```
                {
                        while(T->left!=NULL)
                        {
                                T=T->left;

                        }
                        cout<<T->data;
                }
        int BST:: height(BTNODE *T)
                {
                        int hl,hr;
                        if(T==NULL)
                        {
                                return 0;
                        }
                        if(T->left==NULL && T->right==NULL)
                        {
                                return 0;
                        }
                        hl=height(T->left);
                        hr=height(T->right);
                        if(hl>hr)
                        {
                                return hl+1;
                        }
                        else
                        {
                        return (hr+1);
                       }

                }
        void BST:: maxvalue(BTNODE *T)
                                {
                                        while(T->right!=NULL)
                                        {
                                                T=T->right;

                                        }
                                        cout<<T->data;
                                }
void BST::create()
{
        char ch;
        do
        {
        BTNODE *p=new BTNODE();

         cout<<"\n Enter data ";
         cin>>p->data;
         if(search(root,p->data)==NULL)
         {
                insert(p);
         }
         else
                cout<<"\n The element is already exist";
cout<<"\n If u want to continue if yes press'y' or 'Y'";
```

```
cin>>ch;
}while(ch=='y'||ch=='Y');
}
BTNODE * BST::mirror(BTNODE *T)
{
      BTNODE *p;

      if(T==NULL)
            return NULL;
      if(T!=NULL)
      {
            p=new BTNODE;
            p->data=T->data;
            p->left=mirror(T->right);
            p->right=mirror(T->left);
      }
      return p;
}
void BST::insert(BTNODE *p)
{
       BTNODE *T;
       if(root==NULL)
             root=p;
       else
       {
             T=root;
             while(1)
             {
                   if(p->data < T->data)
                   {
                         if(T->left==NULL)
                         {
                               T->left=p;
                               break;
                         }
                         else
                         T=T->left;
                   }
                   else
                   {

                         if(p->data > T->data)
                         {
                               if(T->right==NULL)
                               {
                                     T->right=p;
                                     break;
                               }
                               else
                               T=T->right;

                         }
                   }
             }
       }
}
```

```
void BST::inorder(BTNODE *T)
{
if(T!=NULL)
{
inorder(T->left);
cout<<" "<<T->data;
inorder(T->right);
}
}
int main()
{
      BST o,o1;
      int choice,T;
      do
      {
      cout<<"\n 1.Create";
      cout<<"\n 2.To find minimum value";
      cout<<"\n 3.To find maximum value";
      cout<<"\n 4.Mirror the tree";
      cout<<"\n 5.Find the no. of nodes in longest path";
      cout<<"\n Enter your choice ";
      cin>>choice;
      switch(choice)
      {
      case 1:
            o.create();
            cout<<"\n The inorder of tree is:";
            o.inorder(o.root);
            break;
      case 2:
            cout<<"\n The minimum value = ";
            o.minvalue(o.root);
            break;
      case 3:
                  cout<<"\n The maximum value = ";
                  o.maxvalue(o.root);
                  break;
      case 4:
                  o1.root=o.mirror(o.root);

                cout<<"\n The inorder of mirror tree is:";
                  o1.inorder(o1.root);
                  break;
      case 5:
            T=o.height(o.root);
            cout<<"\n Longest no of nodes="<<T+1;
            break;
      default :
            cout<<"\n Wrong choice";
            break;
      }
      }while(choice!=5);
      return 0;
}
```