/*Beginning with an empty binary search tree, Construct binary search tree by inserting the values in the order given. (Any Four Operation) After constructing a binary tree -

1. Insert new node
2. Find number of nodes in longest path from root
3. Minimum data value found in the tree
4. Change a tree so that the roles of the left and
5. right pointers are swapped at every node
6. Search a value.

*/

```
#include<iostream>
#include<stdlib.h>
using namespace std;
struct node
{  int a;
  node *left,*right;
 };
class Bt
{
    node *root=NULL,*temp=NULL,*t1=NULL,*s=NULL, *t=NULL;
    int count;
    public:
    Bt(){  count=0;   }
    node *create();
    void insert();
```

```
    void del();
    node *delet(node*,int);
    void find();
    void search();
    void sw();
    void swap(node*);
    void height();
    int he(node*,int);
    void disp(node*);
    void display();
    node *findmin(node*);

};
node *Bt::create()
{
    node *p=new(struct node);
    p->left=NULL;
    p->right=NULL;
    cout<<"\n enter the data";
    cin>>p->a;
    return p;
}
void Bt::insert()
{
    temp=create();
```

```
    if(root==NULL)
    {     root=temp;      }
    else
    {    t1=root;
         while(t1!=NULL)
         {     s=t1;
               if((temp->a)>(t1->a))
               {     t1=t1->right;    }
               else
               {     t1=t1->left;     }
         }
            if((temp->a)>(s->a))
            {     s->right=temp;    }
            else
            {   s->left=temp;      }
     }
}
void Bt::find()
{
   if(root==NULL)
   {    cout<<"\n tree not exist";  }
   else
   {
        t1=root;
```

```
    while(t1->left!=NULL)
    {
            t1=t1->left;
    }

    cout<<"\n smallest no."<<t1->a;
    t1=root;

    while(t1->right!=NULL)
    {
            t1=t1->right;
    }
    cout<<"\n largest no."<<t1->a;
    }
}
void Bt::search()
{
     int m,f=0;
     if(root==NULL)
     {  cout<<"\n tree not exist";
     }
     else
     {
          cout<<"\n enter data to be searched";
          cin>>m;
```

```
        if(root->a==m)
        {   cout<<"\ndata found";   }
        else
        {    t1=root;
             while(t1->a!=m)
             {
                if((m)>(t1->a))
                {    t1=t1->right;  }
                else
                {   t1=t1->left;      }
                if(t1==NULL)
                {   cout<<"\n data not found";     f=1;
                   break;
                }
             }
          if(f==0)
          {   cout<<"\n data found";    }

        }
      }
}
 void Bt::sw()
 {
    if(root==NULL)
    {  cout<<"\n tree not exist";
```

```
    }
    else
    {
      swap(root);
    }
}
void Bt::swap(node *q)
{
  if(q->left!=NULL)
  swap(q->left);
  if(q->right!=NULL)
  swap(q->right);
  t=q->left;
  q->left=q->right;
  q->right=t;
}
void Bt::height()
{
    count=0;
    if(root==NULL)
    {  cout<<"\n tree not exist";
    }
    else
    {
    he(root,0); cout<<"\n height of the tree is"<<count;
```

```
   }
}
int Bt::he(node *q,int c)  // he is a function that will be used to calculate height of the tree. Can be called using root and counter intilized to 0
{
     c++;
    // cout<<"\n*"<<q->a<<"*"<<c<<"*\n";
        if(q->left!=NULL)
     {     he(q->left,c);
     }
        if(q->right!=NULL)
        {
             he(q->right,c);
        }
        if(count<c)
        {
             count=c;
        }



        return 0;
}
void Bt::del()
{    int x;
  cout<<"\n enter data to be deleted";
  cin>>x;
```

```
   delet(root,x);
}
node *Bt::delet(node *T,int x)
{
    if(T==NULL)
    {
       cout<<"\n element not found";
       return(T);
    }
    if(x<T->a)
    {
        T->left=delet(T->left,x);
        return (T);
    }
    if(x>T->a)
    {
         T->right=delet(T->right,x);
         return T;
    }
    if(T->left==NULL&&T->right==NULL)
    {
       temp=T;
       free(temp);
       return(NULL);
    }
```

```
    if(T->left==NULL)
    {
       temp=T;
       T=T->right;
       delete temp;
       return T;
    }
    if(T->right==NULL)
    {
      temp=T;
      T=T->left;
      delete temp;
      return T;
     }
     temp=findmin(T->right);
     T->a=temp->a;
     T->right=delet(T->right,temp->a);
     return T;
}
node *Bt::findmin(node *T)
{
   while(T->left!=NULL)
   {  T=T->left;    }
   return T;
}
```

```
void Bt::display()
{
   if(root==NULL)
   {  cout<<"\n tree not exist";
   }
   else
   {
      disp(root);
   }
}
void Bt::disp(node *q)
{
     cout<<"\n*"<<q->a;
        if(q->left!=NULL)
     {     disp(q->left);
     }
        if(q->right!=NULL)
        {
             disp(q->right);
        }
 }
int main()
{
    Bt b;  int x;     char ch;
    while(1)
```

```
{
        cout<<"\n enter your choice";
        cout<<"\n 1.insert";
        cout<<"\n 2.find";
        cout<<"\n 3.search";
        cout<<"\n 4.swap";
        cout<<"\n 5.height";
        cout<<"\n 6.delete";
        cout<<"\n 7.display";
        cout<<"\n 8.exit";
        cin>>x;
        switch(x)
        {    case 1: b.insert();
                     break;
             case 2: b.find();
                     break;
             case 3: b.search();
                     break;

             case 4: b.sw();
                     break;
             case 5: b.height();
                     break;
             case 6: b.del();
                     break;
```

```
                case 7: b.display();
                        break;
                case 8:exit(0);
            }
        }
    return 0;
}
```