```
/*----------Assignment No - 4----------*/
#include<iostream>
#include<stdio.h>
#include<stdlib.h>
using namespace std;
typedef struct node
{   int a;
    struct node *left,*right;
}node;
struct node *root=NULL;
class BST
{

    int count;
    struct node *temp=NULL,*t1=NULL,*s=NULL, *t=NULL;
    public:
    BST()
    {
        count=0;
    }
    struct node *create();
    void insert();
    int height(struct node*,int c);
    int findmin(struct node*);
    void swap(struct node*);
    void search(struct node * root, int m);
    void display(struct node*);

};

int main()
{
    BST b;
    int x, m, c=0,cnt,min,fl;
    do
    {
        cout<<"\n Enter your choice:";
        cout<<"\n 1.insert:";
        cout<<"\n 2.No of nodes in longest path:";
        cout<<"\n 3.Minimum value:";
        cout<<"\n 4.Swap:";
        cout<<"\n 5.Search:";
        cout<<"\n 6.display:";
        cout<<"\n 7.exit:";
        cin>>x;
        switch(x)
        {
            case 1:
                b.insert();
                break;
            case 2:
                cnt=b.height(root, c);
                cout<<"\n No of nodes in longest path="<<cnt;
                break;
            case 3:
                min=b.findmin(root);
                cout<<"\n Minimum value = "<<min;
                break;
            case 4:
                b.swap(root);
                break;
```

```
            case 5:
                cout<<"\n enter data to be searched:";
                cin>>m;
                b.search(root, m);
                break;
            case 6 :
                b.display(root);
                break;
            case 7:
                exit(0);
        }
    }while(x!=7);
    return 0;
}

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

void BST::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;
        }
    }
}

int BST::height(struct node *q,int c)
{
    if(root==NULL)
```

```
123:     {
124:         cout<<"\n tree not exist";
125:     }
126:     else
127:     {
128:         c++;
129:         if(q->left!=NULL)
130:         {
131:             height(q->left,c);
132:         }
133:         if(q->right!=NULL)
134:         {
135:             height(q->right,c);
136:         }
137:         if(count<c)
138:         {
139:             count=c;
140:         }
141:     }
142:     return count;
143: }
144:
145: int BST::findmin(node *T)
146: {
147:     while(T->left!=NULL)
148:     {
149:         T=T->left;
150:     }
151:     return T->a;
152: }
153:
154: void BST::swap(struct node *q)
155: {
156:     if(q==NULL)
157:     {
158:         cout<<"\n tree not exist";
159:     }
160:     else
161:     {
162:         if(q->left!=NULL)
163:             swap(q->left);
164:         if(q->right!=NULL)
165:             swap(q->right);
166:         t=q->left;
167:         q->left=q->right;
168:         q->right=t;
169:     }
170: }
171:
172: void BST::search(struct node * root, int m)
173: {
174:     int f;
175:     if(root!=NULL)
176:     {
177:         if(root->a==m)
178:             f=1;
179:         if(m>root->a)
180:             search(root->right,m);
181:         else
182:             search(root->left,m);
183:     }
```

```
    if(f==1)
        cout<<"\n FOUND!!!";
}

void BST::display(struct node *q)
{
    if(q==NULL)
    {
        cout<<"\n tree not exist";
    }
    else
    {
        cout<<"\n*"<<q->a;
        if(q->left!=NULL)
        {
            display(q->left);
        }
        if(q->right!=NULL)
        {
            display(q->right);
        }
    }
}
/* OUTPUT

 Enter your choice:
 1.insert:
 2.No of nodes in longest path:
 3.Minimum value:
 4.Swap:
 5.Search:
 6.display:
 7.exit:1

 enter the data56

 Enter your choice:
 1.insert:
 2.No of nodes in longest path:
 3.Minimum value:
 4.Swap:
 5.Search:
 6.display:
 7.exit:1

 enter the data567

 Enter your choice:
 1.insert:
 2.No of nodes in longest path:
 3.Minimum value:
 4.Swap:
 5.Search:
 6.display:
 7.exit:1

 enter the data21

 Enter your choice:
 1.insert:
 2.No of nodes in longest path:
```

```
 3.Minimum value:
 4.Swap:
 5.Search:
 6.display:
 7.exit:3

 Minimum value = 21
 Enter your choice:
 1.insert:
 2.No of nodes in longest path:
 3.Minimum value:
 4.Swap:
 5.Search:
 6.display:
 7.exit:2

 No of nodes in longest path=2
 Enter your choice:
 1.insert:
 2.No of nodes in longest path:
 3.Minimum value:
 4.Swap:
 5.Search:
 6.display:
 7.exit:6

*56
*21
*567
 Enter your choice:
 1.insert:
 2.No of nodes in longest path:
 3.Minimum value:
 4.Swap:
 5.Search:
 6.display:
 7.exit:5

 enter data to be searched:21

 FOUND!!!
 Enter your choice:
 1.insert:
 2.No of nodes in longest path:
 3.Minimum value:
 4.Swap:
 5.Search:
 6.display:
 7.exit:4

 Enter your choice:
 1.insert:
 2.No of nodes in longest path:
 3.Minimum value:
 4.Swap:
 5.Search:
 6.display:
 7.exit:6

*56
*567
```

```
*21
 Enter your choice:
 1.insert:
 2.No of nodes in longest path:
 3.Minimum value:
 4.Swap:
 5.Search:
 6.display:
 7.exit:7

--------------------------------
Process exited after 76.07 seconds with return value 0
Press any key to continue . . .
*/
```