Problem Statement: Beginning with an empty binary search tree, construct a binary search tree by inserting the values in the order given. 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 right pointers are swapped at every node
5. Search a value

```
#include<iostream>
#include<math.h>
using namespace std;

class bnode
{
      int data;
      bnode *left;
      bnode *right;
      bnode *parent;
      friend class bst;
};
class bst{
      public:
            int n,x;
            bnode *root;
            bnode *parent;
            bst()
            {
                  root=NULL;
            }
            bnode *getnode(int ndata)
            {
                  bnode *temp=new bnode;
                  temp->data=ndata;
                  temp->left=NULL;
                  temp->right=NULL;
                  return temp;
            }
      bnode *insert(bnode *temp, int in_data)
      {
            if(temp==NULL){
                  temp=getnode(in_data);
            }
            else if(temp->data>in_data){
                  temp->left=insert(temp->left,in_data);
```

```
	}
	else if(temp->data<in_data)
	{
		temp->right=insert(temp->right,in_data);
	}
	else
	{
		cout<<"Already existing data"<<endl;
	}
	return temp;
}
void input()
{
	cout<<"Enter the number of elements in the BST = ";
	cin>>n;
	for (int i=0;i<n;i++)
	{
		cout<<"Number = ";
		cin>>x;
		root=insert(root,x);
	}
}
void display()
{
	int ch;
	do{
		cout<<endl<<"----Menu----"<<endl;
		cout<<endl<<"Select Option"<<endl;
		cout<<"1.Inorder Traversal"<<endl;
		cout<<"2.Postorder Traversal"<<endl;
		cout<<"3.Preorder"<<endl;
		cout<<"4.Min term"<<endl;
		cout<<"5.Max term"<<endl;
		cout<<"6.Depth of tree"<<endl;
		cout<<"7.Search an element"<<endl;
		cout<<"8.Mirror Tree"<<endl;
		cout<<"9.Exit"<<endl;
		cin>>ch;
		switch(ch){
			case 1:
				inorder(root);
				cout<<endl;
				break;
			case 2:
				postorder(root);
				cout<<endl;
				break;
```

```
                  case 3:
                        preorder(root);
                        cout<<endl;
                        break;
                  case 4:
                        min(root);
                        cout<<endl;
                        break;
                  case 5:
                        maxi(root);
                        cout<<endl;
                        break;
                  case 6:
                        cout<<depth(root)<<endl;
                        break;
                  case 7:
                        int key;
                        cout<<"Enter key to search";
                        cin>>key;
                        search(root,key,parent);
                        cout<<endl;
                        break;
                  case 8:
                        mirror(root);
                        inorder(root);
                        break;
            }
      }while(ch!=9);
}
void inorder(bnode *temp)
{
      if(temp!=NULL)
      {
            inorder(temp->left);
            cout<<temp->data<<" ";
            inorder(temp->right);
      }
}
void postorder(bnode *temp)
{
      if(temp!=NULL)
      {
            postorder(temp->left);
            postorder(temp->right);
            cout<<temp->data<<endl;
      }
}
```

```
void preorder(bnode *temp)
{
      if(temp!=NULL)
      {
            cout<<temp->data<<endl;
            preorder(temp->left);
            preorder(temp->right);
      }

}
void min (bnode *temp){
      while(temp->left !=NULL)
      {
            temp=temp->left;
      }
      cout<<"Minimum Value :"<<temp->data;
}
void maxi(bnode *temp){
      while(temp->right!=NULL)
      {
            temp=temp->right;
      }
      cout<<"Maximum Value :"<<temp->data;
}
int depth(bnode *temp){
      if(temp==NULL){
            return 0 ;
      }
      return(max((depth(temp->left)),depth(temp->right))+1);
}
void mirror(bnode *temp)
{
      if(temp==NULL){
            return;
      }
      else
      {
            bnode *ptr;
            mirror(temp->left);
            mirror(temp->right);
            ptr=temp->left;
            temp->left=temp->right;
            temp->right=ptr;
      }
}
```

```
      void search(bnode *temp,int key,bnode *parent)
      {
            int count=0;
            while(temp!=NULL){
                  if(temp->data ==key)
                  {
                        cout<<"Search after pass : "<<count<<endl;
                        cout<<"\n Element "<<temp->data<<"is present and
its parent is"<<parent->data;

                  }
                  parent=temp;
                  if(temp->data>key)
                  {
                        temp=temp->left;
                  }
                  else
                  {
                        temp=temp->right;
                  }
                  count++;
            }
      }
};
int main(){
      bst obj;
      obj.input();
      obj.display();
      return 0;
}
```

Output

Clear

```
Enter your choice
1.Insert Book
2.Insert Chapter
3.Insert Section
4.Insert Subsection
5.Display Book
6.Exit1

Enter the name: DSA

Enter your choice
1.Insert Book
2.Insert Chapter
3.Insert Section
4.Insert Subsection
5.Display Book
6.Exit2

How many chapters are to be inserted?: 3

Enter the name: Hashing

Enter the name: Trees

Enter the name: Graph

Enter your choice
1.Insert Book
2.Insert Chapter
3.Insert Section
4.Insert Subsection
5.Display Book
6.Exit3

Enter name of chapter in which you want to enter the section: Hashing

How many sections do you want to insert?1
```

Output

```
How many sections do you want to insert?1

Enter the name: Type

Enter your choice
1.Insert Book
2.Insert Chapter
3.Insert Section
4.Insert Subsection
5.Display Book
6.Exit4

Enter name of chapter in which you want to enter the sub section: Hashing

Enter name of section in which you want to enter the sub section: Type

How many sub sections do you want to insert?2

Enter the name: Quadratic

Enter the name: Double

Enter your choice
1.Insert Book
2.Insert Chapter
3.Insert Section
4.Insert Subsection
5.Display Book
6.Exit3

Enter name of chapter in which you want to enter the section: Trees

How many sections do you want to insert?1

Enter the name: BST
```

Output

```
Enter your choice
1.Insert Book
2.Insert Chapter
3.Insert Section
4.Insert Subsection
5.Display Book
6.Exit3

Enter name of chapter in which you want to enter the section: Trees

How many sections do you want to insert?1

Enter the name: BST

Enter your choice
1.Insert Book
2.Insert Chapter
3.Insert Section
4.Insert Subsection
5.Display Book
6.Exit5

Name of Book: DSA
Name of Chapter: Hashing
Name of Section: Type
Name of Sub-section: Quadratic
Name of Sub-section: Double
Name of Chapter: Trees
Name of Section: BST
Name of Chapter: Graph
Enter your choice
1.Insert Book
2.Insert Chapter
3.Insert Section
4.Insert Subsection
5.Display Book
6.Exit
```