Practical-3(B-6)

**Problem Statement:**

Beginning with an empty binary search tree. Construct binary search tree by inserting the values in the order given. After constructing a binary tree -
i. Insert new node
ii. Find number of nodes in longest path from root
iii. Minimum data value found in the tree
iv. Change a tree so that the roles of the left and right pointers are swapped at every node
v. Search a value

**Code:**

```
#include<iostream>

using namespace std;

class node  // Node declaration for BST.

{

   public:

   int data;

   node * left;

   node * right;

};

class BST

{

   public:

   node * root;

   int cnt;


   BST()

   {

      root = NULL;

      cnt=0;

   }


   void insert();

   void inorder(node *temp);

   void smallest();

   void largest();

   int search(int key);

   void Mirror(node *r);

   int height(node *r);

};

void BST::insert()

{
```

```
node *new_node, *temp;
int flag=0;

new_node = new node();  // allocate the memory.
new_node->left = NULL;
new_node->right = NULL;
cout<<"Enter data : ";
cin>>new_node->data;
if(root == NULL)
{
   root = new_node;
}
else
{
   temp = root;
   while(flag ==0)
   {
      if(new_node->data < temp->data)
      {
         if(temp->left==NULL)
         {
            temp->left = new_node;
            flag=1;
         }
         else
         {
            temp= temp->left;
         }
      }
      else if(new_node->data > temp->data)
      {
         if(temp->right == NULL)
         {
            temp->right = new_node;
            flag =1;
         }
         else
         {
            temp = temp -> right;
```

```
            }
        }
        else
        {
            cout<<"\nData is already exist in the Tree";
            flag++;
        }
    }
  }
}

void BST:: inorder(node * temp)
{
  if(temp != NULL)
  {
    inorder(temp->left);
    cout<<" "<<temp->data;
    cnt++;
    inorder(temp->right);
  }
}
void BST :: smallest()
{
  node *temp;
  temp = root;
  while(temp ->left !=NULL)
  {
    temp = temp ->left;
  }

  cout<<"\nSmallest node in the tree is : "<<temp->data;


}
void BST :: largest()
{
  node *temp;
  temp = root;
  while(temp ->right !=NULL)
  {
```

```
        temp = temp ->right;
    }

    cout<<"\nLargest5 node in the tree is : "<<temp->data;

}

int BST :: search(int key)
{
    node *temp;
    temp = root;

    while(1)
    {
        if(key < temp->data)
        {
            if(temp->left !=NULL)
            {
                temp=temp->left;
            }
            else
            {
                return(0);
            }
        }
        else if(key > temp->data)
        {
            if(temp->right != NULL)
            {
                temp=temp->right;
            }
            else
            {
                return(0);
            }
        }
        else
        {
            return(1);
```

```
        }
    }
}

void BST::Mirror(node *root)
{
    node *temp;
    if(root != NULL)
    {
        temp = root->left;
        root->left = root->right;
        root->right = temp;
        Mirror(root->left);
        Mirror(root->right);

    }
}
int BST::height(node *r)
{
    int Left_Height,Right_Height;
    if(r == NULL)
        return(0);
    if(r->left == NULL && r->right == NULL)
        return(0);
     Left_Height = height(r->left);
    Right_Height = height(r->right);
    if(Left_Height > Right_Height)
        return(Left_Height +1);
    else
        return(Right_Height +1);
}
int main()
{
    int ch;
    BST B;
    node *test;
    node test2;
    test = new node();
    do{
```

```
cout<<"\n Menu";
cout<<"\n1. Insert";
cout<<"\n2. Inorder";
cout<<"\n3. smallest number";
cout<<"\n4. Largest number";
cout<<"\n5. Search";
cout<<"\n6. Mirror";
cout<<"\n7. Height of the tree";
cout<<"\n8. Exit";
cout<<"\nEnter your choice : ";
cin>>ch;
switch(ch)
{
   case 1 :  //Insert operation.
      B.insert();
   break;
   case 2 :  //Inorder display.
      B.inorder(B.root);
      cout<<"\nCnt = "<<B.cnt;
   break;
   case 3 :  //smallest number in BST.
      B.smallest();
   break;
   case 4 :  //largest number in BST.
      B.largest();
   break;
   case 5 :  //Search a number in BST.
      int key;
      cout<<"Enter number to Search : ";
      cin>>key;
      if(B.search(key))
      {
         cout<<"number is found";
      }
      else
      {
         cout<<"number is not found";
      }
   break;
```

```
        case 6 :  //Mirror image of BST
            cout<<"\nBefore Mirror Image :";
            B.inorder(B.root);
            B.Mirror(B.root);
            cout<<"\nAfter Mirror Image :";
            B.inorder(B.root);
        break;
        case 7 :  //Height of the BST.
            cout<<"Height of the BSt is " <<B.height(B.root);
        break;
        case 8 : cout<<"----Exit----";
            break;
        default :
            cout<<"\nEnter correct choice ";
        break;
      }
   }while(ch !=8);
   return(1);
}
```

**OUTPUT:**

```
Menu
1. Insert
2. Inorder
3. smallest number
4. Largest number
5. Search
6. Mirror
7. Height of the tree
8. Exit
Enter your choice : 1
Enter data : 8
Menu
1. Insert
2. Inorder
3. smallest number
4. Largest number
5. Search
6. Mirror
7. Height of the tree
```

8. Exit

Enter your choice : 1

Enter data : 4

Menu

1. Insert

2. Inorder

3. smallest number

4. Largest number

5. Search

6. Mirror

7. Height of the tree

8. Exit

Enter your choice : 1

Enter data : 9

Menu

1. Insert

2. Inorder

3. smallest number

4. Largest number

5. Search

6. Mirror

7. Height of the tree

8. Exit

Enter your choice : 1

Enter data : 3

Menu

1. Insert

2. Inorder

3. smallest number

4. Largest number

5. Search

6. Mirror

7. Height of the tree

8. Exit

Enter your choice : 2

3 4 8 9

Cnt = 4

Menu

1. Insert

2. Inorder

3. smallest number

4. Largest number

5. Search

6. Mirror

7. Height of the tree

8. Exit

Enter your choice : 3

Smallest node in the tree is : 3

Menu

1. Insert

2. In order

3. smallest number

4. Largest number

5. Search

6. Mirror

7. Height of the tree

8. Exit

Enter your choice : 4

Largest5 node in the tree is : 9

Menu

1. Insert

2. Inorder

3. smallest number

4. Largest number

5. Search

6. Mirror

7. Height of the tree

8. Exit

Enter your choice: 5

Enter number to Search: 9

number is found

Menu

1. Insert

2. Inorder

3. smallest number

4. Largest number

5. Search

6. Mirror

7. Height of the tree

8. Exit

Enter your choice : 5

Enter number to Search : 2

number is not found

Menu

1. Insert

2. Inorder

3. smallest number

4. Largest number

5. Search

6. Mirror

7. Height of the tree

8. Exit

Enter your choice : 6

Before Mirror Image : 3 4 8 9

After Mirror Image : 9 8 4 3

Menu

1. Insert

2. Inorder

3. smallest number

4. Largest number

5. Search

6. Mirror

7. Height of the tree

8. Exit

Enter your choice : 7

Height of the BSt is 2

Menu

1. Insert

2. Inorder

3. smallest number

4. Largest number

5. Search

6. Mirror

7. Height of the tree

8. Exit

Enter your choice : 8

----Exit----