**Assignment number: 1**

**Subject: ADVANCED DATA STRUCTURES LAB**

Name: ***RIA MITTAL***

Class: ***SECOND YEAR ENGINEERING***

Division: ***B***

Roll no: ***222008***

Batch: ***B1***

**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

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>

#include<string>

using namespace std;

int validate(int a)

{

cin>>a;

while(cin.fail())

{

cin.clear();

cout<<"Invalid Input !!\nEnter a Integer Value : ";

cin>>a;

}

return a;

}

class node

{

friend class tree;

int data;

node \*left,\*right;

};

class tree

{

friend class node;

node \*root;

public :

tree()

{

root=NULL;

New=NULL;

}

node \*New;

node \*create(int d,node \*root);

void display(node \*root);

void search(node \*root,int d);

void insert();

node \*leftMost(node \*);

node \* mirror(node \*);

node \*getRoot() {

return this->root;

}

}bst;

node \*tree :: create(int d,node \*root)

{

if(root==NULL)

{

New=new node;

New->data=d;

New->left=NULL;

New->right=NULL;

root=New;

}

else {

if(root->data > d)

{

root->left=create(d,root->left);

}

else if(root->data < d)

{

root->right=create(d,root->right);

}

else

{

cout<<"Key already Present . . \n";

}

}

return root;

}

void tree :: display(node \*root)

{

if(root->left!=NULL) display(root->left);

cout<<root->data<<" ";

if(root->right!=NULL) display(root->right);

}

void tree :: search(node \*root,int d)

{

if(root!=NULL)

{

if(root->data==d)

{

cout<<"Data "<<root->data<<" is present in your BST !!\n";

}

else if(root->data > d) search(root->left,d);

else if(root->data < d) search(root->right,d);

}

else cout<<"NOT FOUND\n";

}

void tree :: insert()

{

int d;

char ch;

do {

cout<<"Enter the data to be inserted : ";

d=validate(d);

root=create(d,root);

cout<<"Enter Another Node ? ";

cin>>ch;

} while(ch=='y');

}

node\* tree :: leftMost(node \*temp)

{

for(;temp->left!=NULL;temp=temp->left) ;

cout<<temp->data<<endl;

}

node \*tree :: mirror(node \*root)

{

node \*temp=NULL;

if(root!=NULL)

{

temp = root->left;

root->left = root->right;

root->right = temp;

mirror(root->left);

mirror(root->right);

}

return root;

}

int main()

{

char ch;

int d,op;

do {

cout<<"1.Insert\n2.Search\n3.Display\n4.Lowest Element\n5.Mirror\n6.Exit\n";

cin>>op;

switch(op)

{

case 1:

{

bst.insert();

break;

}

case 2:

{

cout<<"Enter the value to be searched : ";

cin>>d;

bst.search(bst.getRoot(),d);

break;

}

case 3:

{

bst.display(bst.getRoot());

cout<<endl;

break;

}

case 4 :

{

cout<<"The Lowest Element in your BST is : ";

bst.leftMost(bst.getRoot());

break;

}

case 5:

{

bst.mirror(bst.getRoot());

break;

}

default :

{

return 0;

}

}

} while(1);

return 0;

}

OUTPUT:

1.Insert

2.Search

3.Display

4.Lowest Element

5.Mirror

6.Exit

1

Enter the data to be inserted : 3

Enter Another Node ? y

Enter the data to be inserted : 7

Enter Another Node ? y

Enter the data to be inserted : 4

Enter Another Node ? y

Enter the data to be inserted : 66

Enter Another Node ? y

Enter the data to be inserted : 11

Enter Another Node ? y

Enter the data to be inserted : 88

Enter Another Node ? y

Enter the data to be inserted : 66

Key already Present . .

Enter Another Node ? y

Enter the data to be inserted : 33

Enter Another Node ? y

Enter the data to be inserted : 14

Enter Another Node ? y

Enter the data to be inserted : 14

Key already Present . .

Enter Another Node ? n

1.Insert

2.Search

3.Display

4.Lowest Element

5.Mirror

6.Exit

3

3 4 7 11 14 33 66 88

1.Insert

2.Search

3.Display

4.Lowest Element

5.Mirror

6.Exit

2

Enter the value to be searched : 11

Data 11 is present in your BST !!

1.Insert

2.Search

3.Display

4.Lowest Element

5.Mirror

6.Exit

4

The Lowest Element in your BST is : 3

1.Insert

2.Search

3.Display

4.Lowest Element

5.Mirror

6.Exit

5

1.Insert

2.Search

3.Display

4.Lowest Element

5.Mirror

6.Exit

3

88 66 33 14 11 7 4 3

1.Insert

2.Search

3.Display

4.Lowest Element

5.Mirror

6.Exit

5

1.Insert

2.Search

3.Display

4.Lowest Element

5.Mirror

6.Exit

1

Enter the data to be inserted : 40

Enter Another Node ? y

Enter the data to be inserted : 77

Enter Another Node ? n

1.Insert

2.Search

3.Display

4.Lowest Element

5.Mirror

6.Exit

3

3 4 7 11 14 33 40 66 77 88

1.Insert

2.Search

3.Display

4.Lowest Element

5.Mirror

6.Exit

6