**Assignment No : 3**

**Aim:**

Construct binary search tree by inserting the values in the order given. After constructing a binary search tree

1.Insert new node

**Objective**: **:**

Understand the problem statement, determine and implement the various

constructions on BST .

Understand the use of Stack for BST insertion.

**Theory**:

Binary search tree is a binary tree in which every node satisfies the following conditions:

* + All values in the left sub tree of a node are less than the value of the node.
  + All values in the right sub tree of a node are greater than the value of the node.
  + Every node has a key and no two elements have same keys

The left and right sub trees are also binary search tree

200px-Binary_search_tree.svg

Insertion:

* The way to insert a new node in the tree, its value is first compared with the value of the root. If its value is less than the root's, it is then compared with the value of the root's left child. If its value is greater, it is compared with the root's right child. This process continues, until the new node is compared with a leaf node, and then it is added as this node's right or left child, depending on its value.
* Another way is examine the root and recursively insert the new node to the **left sub tree**if the new value is less than or equal to the root, or the **right sub tree**if the new value is greater than the root.

Example:

100 100

/ \ Insert 40 / \

20 500 ---------> 20 500

/ \ / \

10 30 10 30

\

40

**Algorithm:**

To insert a new value *v* into a binary search tree *T*, we use the procedure TREE-INSERT. The procedure is passed a node *z* for which *key*[*z*] = *v*, *left*[*z*] = NIL, and *right*[*z*] = NIL. It modifies *T* and some of the fields of *z* in such a way that *z* is inserted into an appropriate position in the tree.

TREE-INSERT(*T*,*z*)

1 y http://staff.ustc.edu.cn/~csli/graduate/algorithms/images/arrlt12.gif NIL

2 x http://staff.ustc.edu.cn/~csli/graduate/algorithms/images/arrlt12.gif root[T]

3 while x http://staff.ustc.edu.cn/~csli/graduate/algorithms/images/noteq.gif NIL

4 do y http://staff.ustc.edu.cn/~csli/graduate/algorithms/images/arrlt12.gif x

5 if key[z] < key[x]

6 then x http://staff.ustc.edu.cn/~csli/graduate/algorithms/images/arrlt12.gif left[x]

7 else x http://staff.ustc.edu.cn/~csli/graduate/algorithms/images/arrlt12.gif right[x]

8 p[z] http://staff.ustc.edu.cn/~csli/graduate/algorithms/images/arrlt12.gif y

9 if y = NIL

10 then root[T] http://staff.ustc.edu.cn/~csli/graduate/algorithms/images/arrlt12.gif z

11 else if key[z] < key[y]

12 then left[y] http://staff.ustc.edu.cn/~csli/graduate/algorithms/images/arrlt12.gif z

13 else right[y] http://staff.ustc.edu.cn/~csli/graduate/algorithms/images/arrlt12.gif z

**Program:**

**/\***

**Problem Statement : Construct binary search tree by inserting the values in the order given. After constructing a binary search tree**

**i.Insert new node**

**ii. Find number of nodes in longest path along with its height**

**iii. Minimum data value found in the tree**

**\*/**

#include<iostream>

#include<math.h>

using namespace std;

struct tnode{

int data;

tnode \*lptr,\*rptr;

};

tnode \*root=NULL;

tnode \*insert(int val);

void create(int val);

void inorder(tnode \*);

void preorder(tnode \*);

void postorder(tnode\*);

int minimum(tnode \*);

int height(tnode \* );

int main()

{

int num,a,choise;

cout<<"Nodes in tree : ";

cin>>a;

for(int i=0;i<a;i++)

{

cout<<"Enter the data : ";

cin>>num;

create(num);

}

do {

cout<<"-------------------------------------------------------------------------\n";

cout<<"\n\t\t\tHave Fun with Binary Search Tree!!\n";

cout<<"\n\t\t1.Inser a new node\n\t\t2.Find heiht of tree\n\t\t3.Minimum node present in Tree \n\t\t4.Inorder Traversal\n\t\t5.Exit\n\t\tYour choise : ";

cin>>choise;

switch(choise)

{

case 1:

cout<<"Enter the data : ";

cin>>num;

create(num);

cout<<"\tNode inserted Successflly in Tree!!\n";

break;

case 2:

cout<<"The Height of tree is : ";

cout<<height(root)<<endl;

break;

case 3:

cout<<"The minimum of tree is : ";

cout<<minimum(root)<<endl;

break;

case 4:

cout<<"Inorder Traversal of Tree : \n";

inorder(root);

break;

}

}

while(choise<5);

}

tnode \*insert(int val)

{

tnode \*newNode =new tnode;

newNode->data=val;

newNode->lptr =newNode->rptr =NULL;

return newNode;

}

void create(int val)

{

tnode \*parent,\*current=root;

if(root==NULL)

{

root = insert(val);

}

else

{

while(current!=NULL)

{

parent =current;

if(val<current->data)

current =current->lptr;

else

current = current->rptr;

}

if(val<parent->data)

parent->lptr=insert(val);

else

parent->rptr =insert(val);

}

}

void inorder(tnode \*Root)

{

tnode \*temp =Root;

if(temp==NULL)

return ;

inorder(temp->lptr);

cout<<temp->data<<endl;

inorder(temp->rptr);

}

void preorder(tnode \*Root)

{

tnode \*temp=Root;

if(temp==NULL)

return ;

cout<<temp->data<<endl;

preorder(temp->lptr);

preorder(temp->rptr);

}

void postorder(tnode \*Root)

{

tnode \*temp =Root;

if(temp==NULL)

return ;

preorder(temp->lptr);

preorder(temp->rptr);

cout<<temp->data<<endl;

}

int minimum(tnode \*Root)

{

tnode \*parent,\*temp = Root;

while(temp!=NULL)

{

parent = temp ;

temp =temp->lptr;

}

return parent->data;

}

int height(tnode \*Root)

{

tnode \*current =Root;

if(current == NULL)

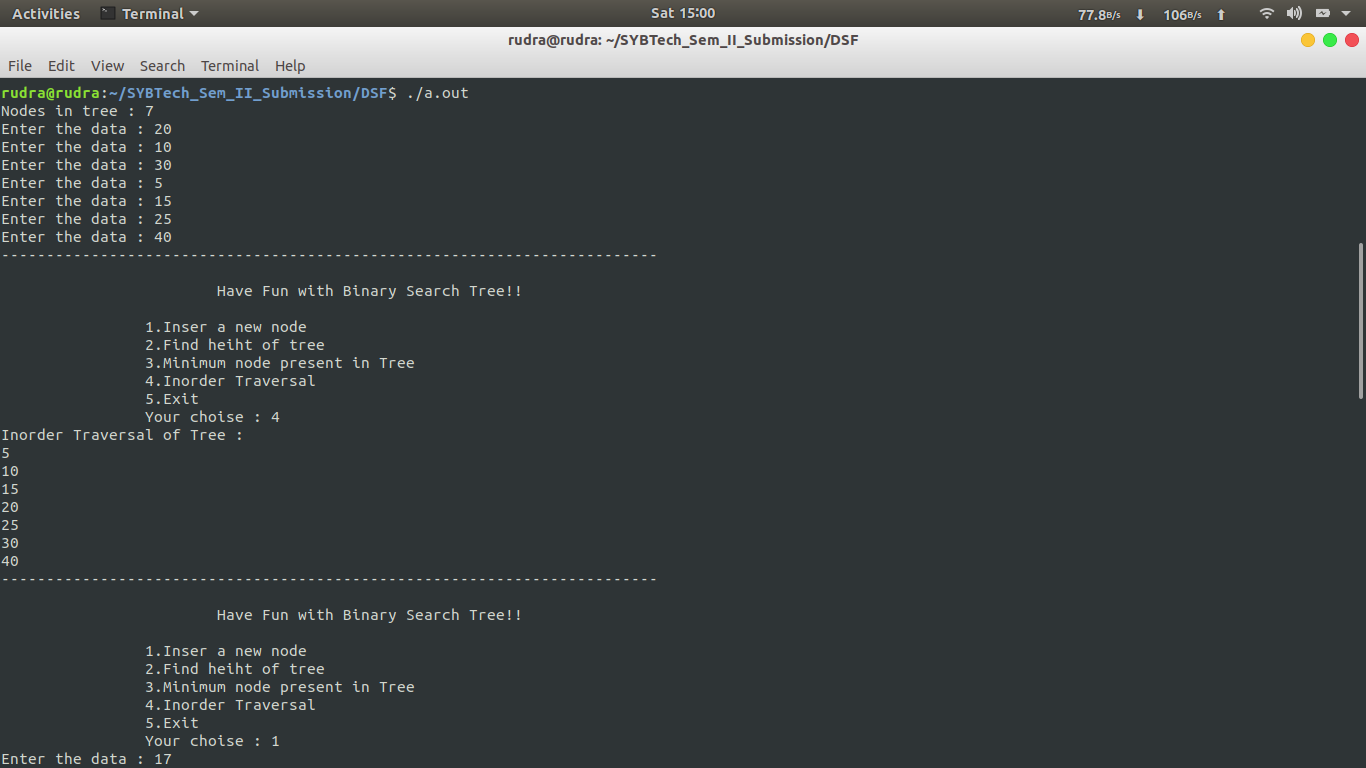
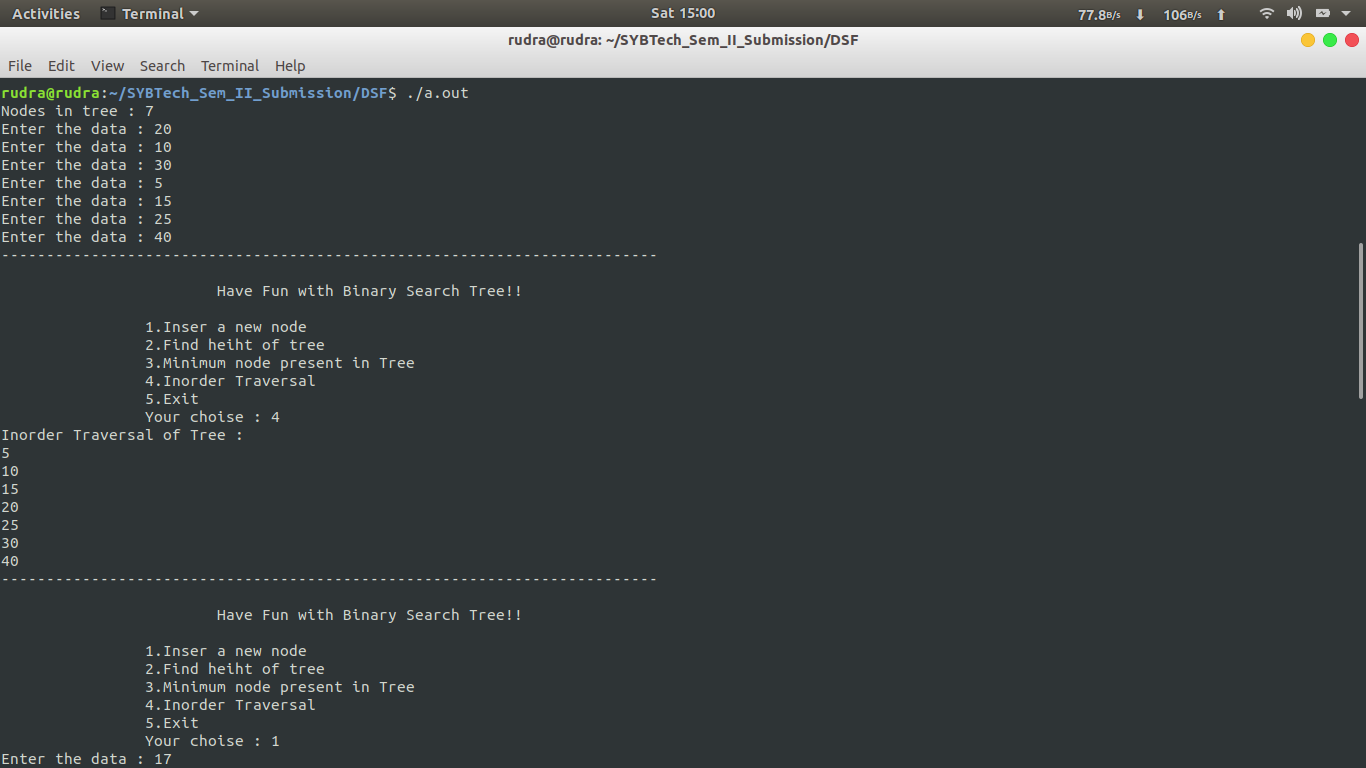
return 0;

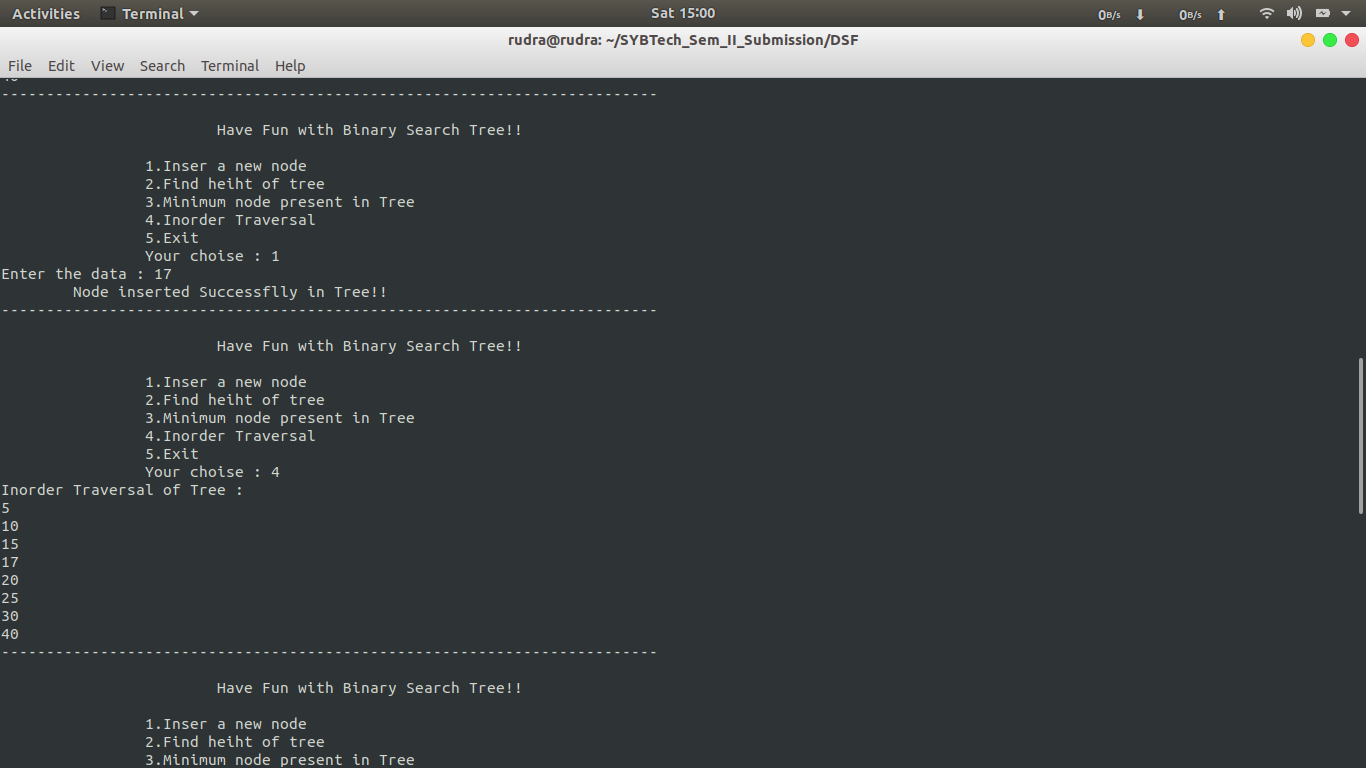
int ht = max(height(current->lptr),height(current->rptr));

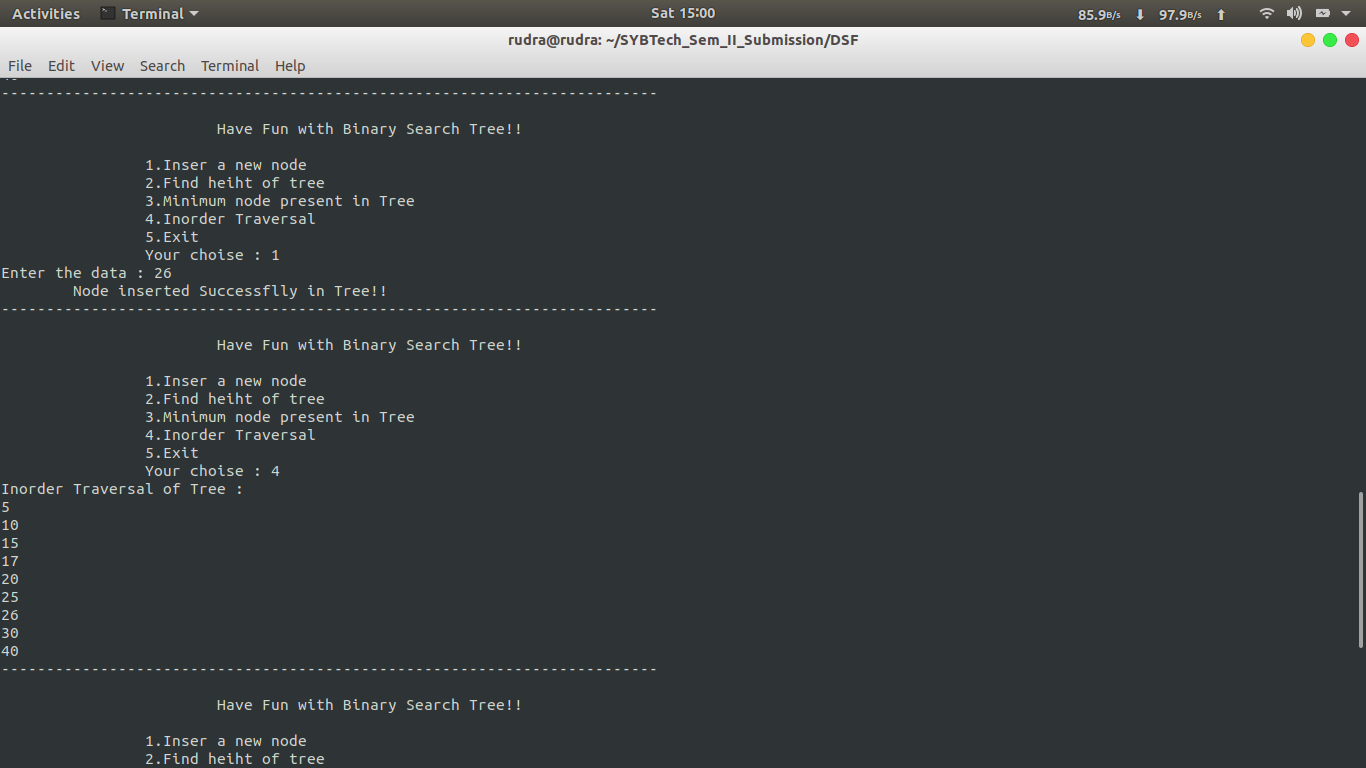
return ht+1;

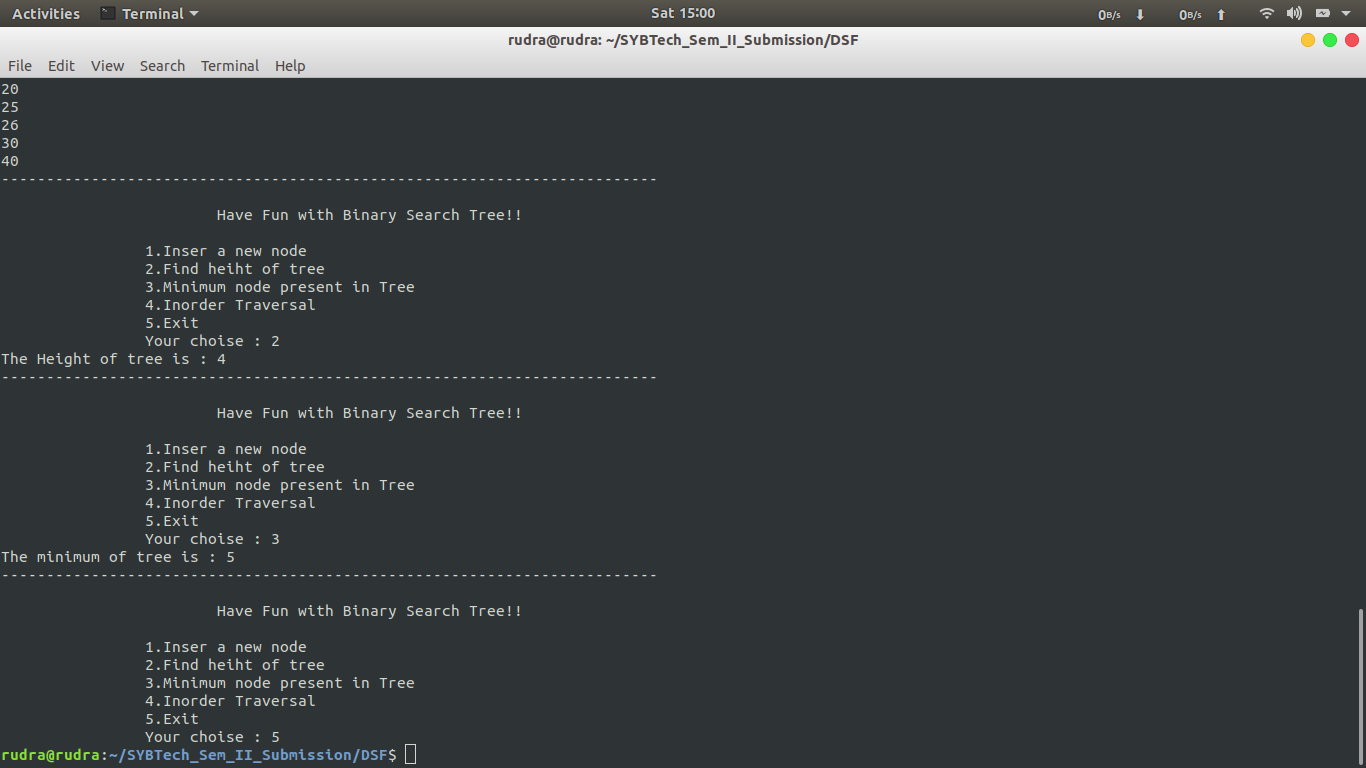
}

**Output:**









**Conclusion:** In this assignment, we have learnt the construction of binary search tree and implementation of insertion.