**Assignment 4**

**Aim:**

Modify the BST such that the roles of the left and right pointers are swapped at every node

**Objective :**

Swapping left and right node of the BST

**Theory:**

Two of the nodes of a Binary Search Tree (BST) are swapped. Fix (or correct) the BST.

Input Tree:

10

/ \

5 8

/ \

2 20

In the above tree, nodes 20 and 8 must be swapped to fix the tree.

Following is the output tree

10

/ \

5 20

/ \

2 8

The inordertraversal of a BST produces a sorted array. So a **simple method** is to store inorder traversal of the input tree in an auxiliary array. Sort the auxiliary array. Finally, insert the auxiilary array elements back to the BST, keeping the structure of the BST same. Time complexity of this method is O(nLogn) and auxiliary space needed is O(n).

**1.** The swapped nodes are not adjacent in the inorder traversal of the BST.

For example, Nodes 5 and 25 are swapped in {3 5 7 8 10 15 20 25}.

The inorder traversal of the given tree is 3 25 7 8 10 15 20 5

If we observe carefully, during inorder traversal, we find node 7 is smaller than the previous visited node 25. Here save the context of node 25 (previous node). Again, we find that node 5 is smaller than the previous node 20. This time, we save the context of node 5 ( current node ). Finally swap the two node’s values.

**Algorithm:**

(1) Call swap for left-subtree i.e., swap(left-subtree)

(2) Call swap for right-subtree i.e., swap(right-subtree)

(3) Swap left and right subtrees.

temp = left-subtree

left-subtree = right-subtree

right-subtree = temp

**Code :**

**Problem Statement : Modify the above BST such that the roles of the left and right pointers are swapped at every node**

#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 swapTree(tnode \*);

void swap(tnode \*,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-------------------------------------------------------------------------\n";

cout<<"\n\t\t\tPlay With Binary Search Tree\n";

cout<<"\n\t\t1.Inser a new node\n\t\t2.Swipe the tree\n\t\t3.Inorder traversal\n\t\t4.Exit\n\t\tYour choise : ";

cin>>choise;

switch(choise)

{

case 1:

cout<<"Enter the data : ";

cin>>num;

create(num);

break;

case 2:

swapTree(root);

cout<<"Tree Swiped ! ";

break;

case 3:

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

inorder(root);

break;

}

}while(choise<4);

}

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);

}

int swapTree(tnode \*Root)

{

tnode \*current = Root;

if(current==NULL)

return 0;

tnode \*temp =current->rptr;

current->rptr = current->lptr;

current->lptr = temp;

//swap(current->lptr,current->rptr);

swapTree(current->lptr);

swapTree(current->rptr);

}

void swap(tnode \*r,tnode \*l)

{

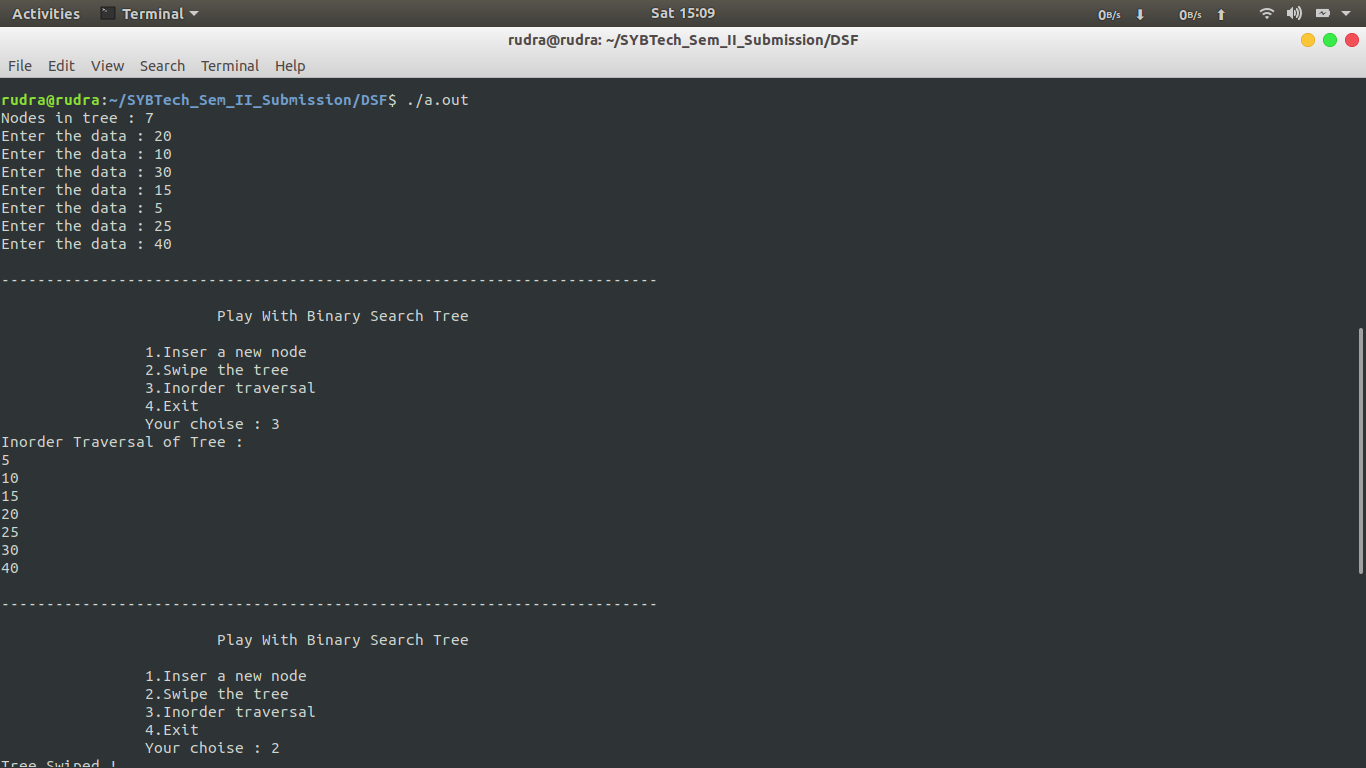
tnode \*temp = r;

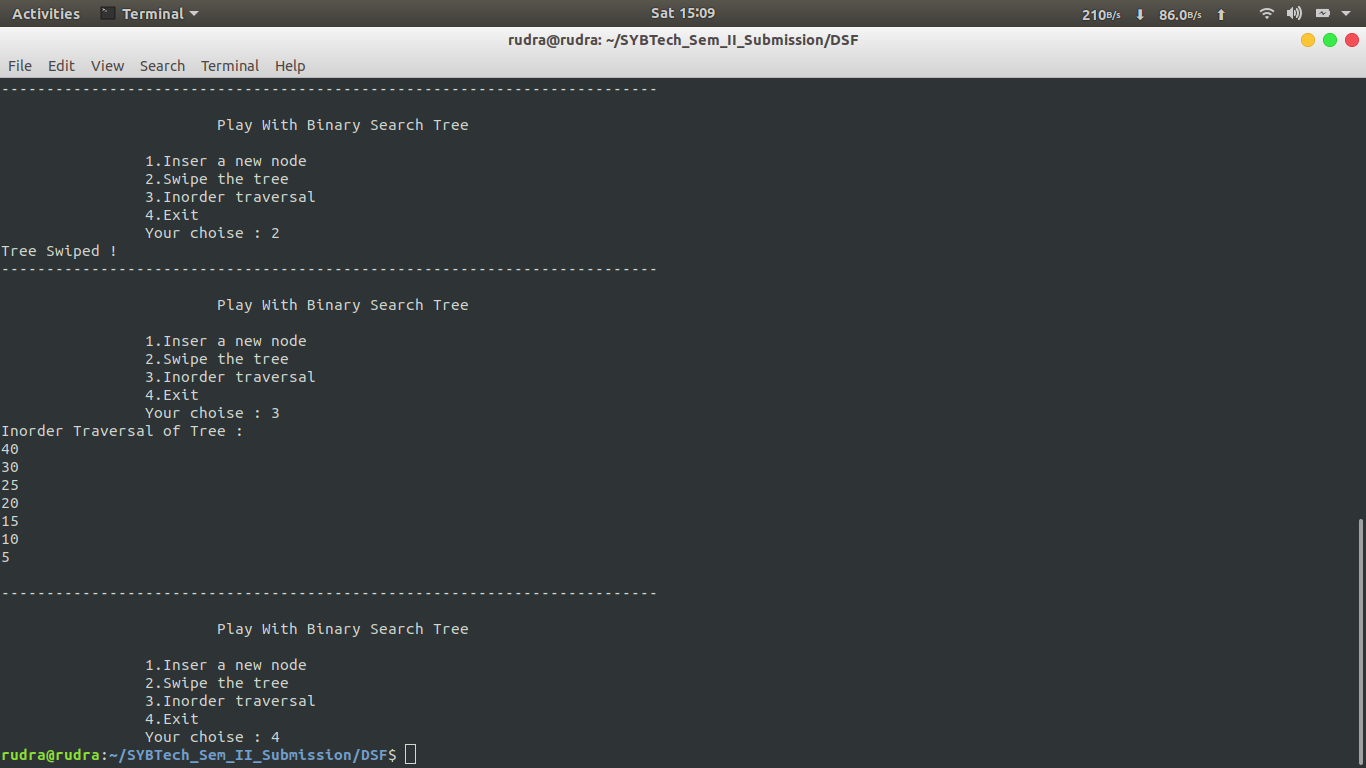
r = l;

l = temp;

}

**Output:**





**Conclusion:**

This assignment is used how to swap left and right node of the BST