**Name:Prathamesh Sharad Hande**

**Class:SE Comp.**

**Seat No.:S191194223**

**PRN No:72222456H**

**4)Binary Search:To implement the operation on binary search tree.**

**Beginning with an empty binary search tree, Construct binary search tree by inserting**

**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**

#include<iostream>

#include<stdlib.h>

using namespace std;

class bstnode

{

public:

int data;

bstnode\*left,\*right;

bstnode(int x)

{

data=x;

left=right=NULL;

}

};

class bst

{

bstnode\*root;

public:

bst()

{

root=NULL;

}

bstnode\*create();

void insert(int x);

bstnode\*find(int x);

bstnode\*find\_min(bstnode\*root);

int longest\_path(bstnode\*T);

void display(bstnode\*t);

bstnode\*swapper(bstnode\*t);

};

bstnode\*bst::create()

{

int x,i,n;

root=NULL;

cout<<"enter total number of nodes :";

cin>>n;

cout<<"enter tree value :";

for(i=0;i<n;i++)

{

cin>>x;

insert(x);

}

return(root);

}

void bst::insert(int x)

{

bstnode\*p,\*q,\*r;

r=new bstnode (x);

if(root==NULL)

{

root=r;

return;

}

p=root;

while(p!=NULL)

{

q=p;

if(x>p->data)

p=p->right;

else

p=p->left;

}

if(x>q->data)

q->right=r;

else

q->left=r;

}

bstnode\*bst::find(int x)

{

while(root!=NULL)

{

if(x==root->data)

return (root);

if(x>root->data)

root=root->right;

else

root=root->left;

}

return NULL;

}

bstnode \*bst::find\_min(bstnode\*root)

{

while(root->left!=NULL)

{

return(root->left);

}

return(root);

}

int bst::longest\_path(bstnode\*T)

{

int hl,hr;

if(T==NULL)

return(0);

if(T->left==NULL && T->right==NULL)

return(0);

hl=longest\_path(T->left);

hr=longest\_path(T->right);

if(hl>hr)

{

return(hl+1);

}

else

{

return(hr+1);

}

}

void bst::display(bstnode\*t)

{

if(t!=NULL)

{

display(t->left);

cout<<"\t"<<t->data;

display(t->right);

}

}

bstnode\*swapper(bstnode\*t)

{

bstnode\*c;

if(t!=NULL)

{

c=t->left;

t->left=swapper(t->right);

t->right=swapper(c);

}

return(t);

}

int main()

{

int ch,x,i;

bst b;

bstnode\*p,\*q,\*root;

do

{

cout<<"\n1.create \n2.find \n3.find\_min\n4.longest\_path\n5.display\n6.swap";

cout<<"\nenter u r choice : ";

cin>>ch;

switch(ch)

{

case 1:

root=b.create();

break;

case 2:

cout<<"enter node to be searched ";

cin>>x;

p=b.find(x);

if(p==NULL)

cout<<"\nnode not found ";

else

cout<<"node found"<<p->data;

break;

case 3:

q=b.find\_min(root);

cout<<"minimum value in tree "<<q->data;

break;

case 4:

i=b.longest\_path(root);

cout<<" longest path in tree "<<i+1;

break;

case 5:

b.display(root);

break;

case 6:

swapper(root);

break;

}

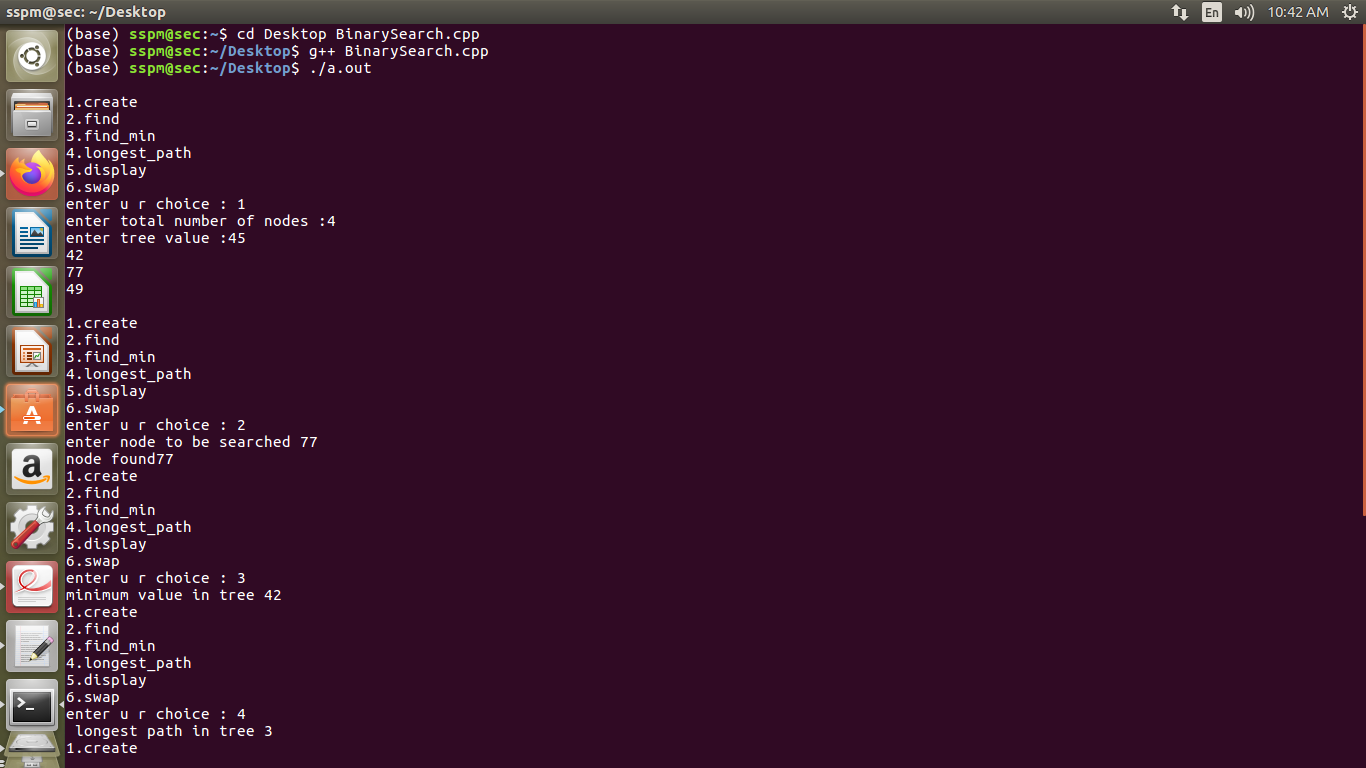
}

while(ch!=7);

return 0;}

**Output:**

1)



2)

