**NAME** : Tejesh Santosh Yewale

**ROLL NO. :** A-61

**PRACTICAL NO. B2**

**CODE:**

#include<iostream>

using namespace std;

struct node

{

int data;

node \*left,\*right;

};

class BST

{

public:

node \*root;

BST()

{

root=NULL;

}

node \*create\_BST();

node \* insert(node \*, int);

void display(node \*);

void mirror(node \*);

node \* findmin(node \*);

node \* findmax(node \*);

node \* find(node \*, int );

int height(node \*);

};

node \* BST::create\_BST()

{

int n, i, x;

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

cin>>n; //7

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

{

cout<<"Enter Data : ";

cin>>x;

root=insert(root,x);

}

return root;

}

node \* BST ::insert(node \* T, int x)

{

if(T==NULL)

{

T=new node;

T->data=x;

T->left=NULL;

T->right=NULL;

return T;

}

else

{

if(x>T->data)

{

T->right=insert(T->right,x);

return T;

}

if(x<T->data)

{

T->left=insert(T->left,x);

return T;

}

}

}

void BST::display(node \* T)

{

if(T!=NULL)

{

display(T->left);

cout<<T->data<<" ";

display(T->right);

}

}

void BST::mirror(node \* T)

{

if(T==NULL){

return;

}

else{

node\* Temp=new node;

mirror(T->left);

mirror(T->right);

Temp=T->left;

T->left=T->right;

T->right=Temp;

}

}

node \* BST::findmin(node \* T)

{

while(T->left!=NULL)

{

T=T->left;

}

return T;

}

node \* BST::findmax(node \* T)

{

while(T->right!=NULL)

{

T=T->right;

}

return T;

}

node \* BST::find(node \*T, int x)

{

if(T==NULL)

return NULL;

if(T->data==x)

{

cout<<"found"<<T->data;

return T;

}

if (x > T->data )

{

T=find(T->right,x);

return T;

}

else if (x < T->data)

{

T=find(T->left,x);

return T;

}

}

int BST::height(node \*T)

{

if(T==NULL)

return 0;

else

return max(height(T->left), height(T->right))+1;

}

int main()

{

BST b1;

int ch,x1;

char ans;

int key;

node \*temp,\*temp1;

do

{

cout<<" \n MAIN MENU \n 1.Create \n 2.Insert \n 3.Display \n 4.Findmin \n 5.Findmax \n 6.Find \n 7.Height \n 8.Swap";

cout<<"\n\n Enter your choice : ";

cin>>ch;

switch(ch)

{

case 1: b1.root=b1.create\_BST();

break;

case 2:

cout<<"Enter Data : ";

cin>>x1;

b1.root=b1.insert(b1.root,x1);

break;

case 3:

cout<<"THE BST TREE in INORDER : \n ";

b1.display(b1.root);

break;

case 4: temp=b1.findmin(b1.root);

cout<<"\n Min value from BST is :"<<temp->data;

break;

case 5: temp=b1.findmax(b1.root);

cout<<"\n Max value from BST is :"<<temp->data;

break;

case 6: cout<<" \n Enter Key value to be search : ";

cin>>key;

temp1=b1.find(b1.root,key);

if(temp1!=NULL)

cout<<"Present "<<temp1->data;

else

cout<<"\n Key is not Present in BST ";

break;

case 7: x1=b1.height(b1.root);

cout<<" Height of BST tree is : "<<x1-1;

break;

case 8: cout<<"Before Swapped \n ";

cout<<"\nTHE BST TREE in INORDER : \n ";

b1.display(b1.root);

b1.mirror(b1.root);

cout<<"\nAfter Swapped \n ";

cout<<"\nTHE BST TREE in INORDER : \n ";

b1.display(b1.root);

}

cout<<"\n \n Go to Main Menu ??(y or n) : ";

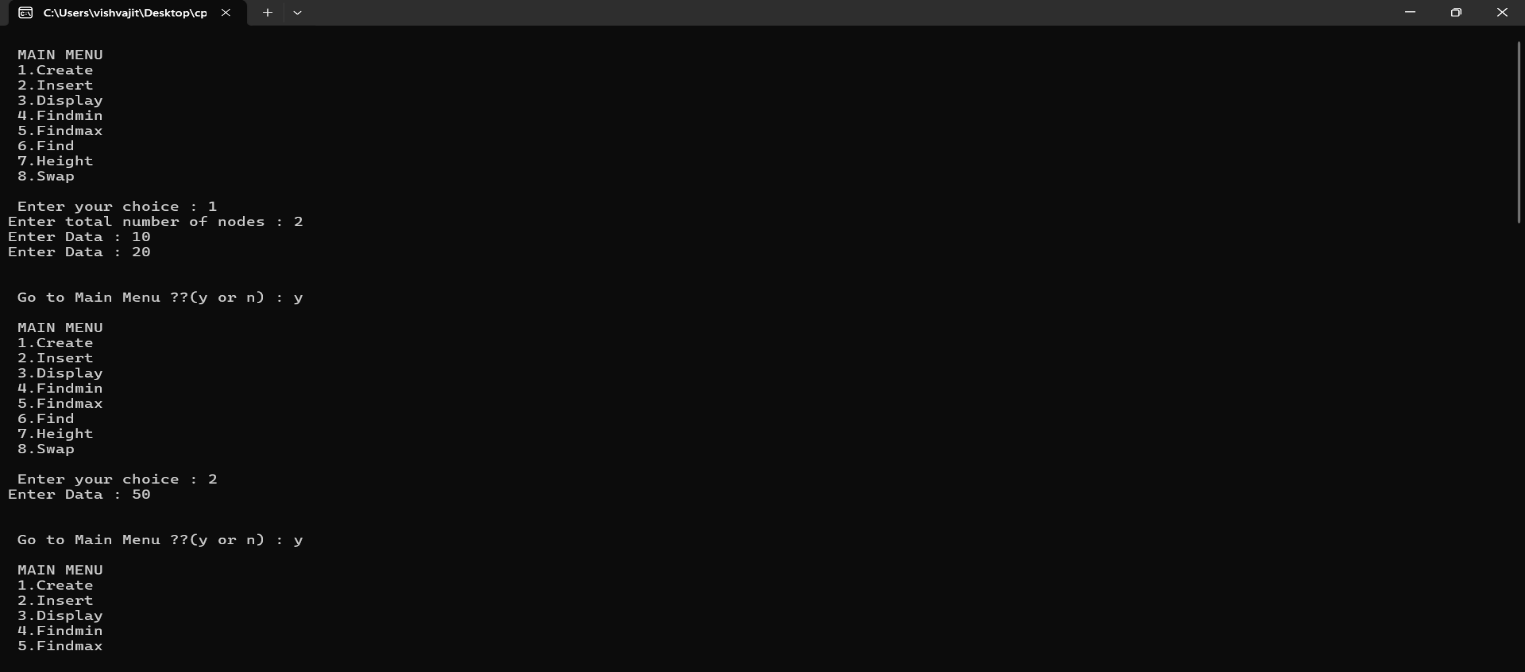
cin>>ans;

}while(ans=='y');

return 0;

}

**OUTPUT:**

****

