#include<iostream>

using namespace std;

struct tree\_node

{

tree\_node \*left;

tree\_node \*right;

int data;

} ;

class bst

{

tree\_node \*root;

public:

bst()

{

root=NULL;

}

int isempty()

{

return(root==NULL);

}

void insert(int item);

void inordertrav();

void inorder(tree\_node \*p);

void postordertrav();

void postorder(tree\_node \*);

void preordertrav();

void preorder(tree\_node \*);

};

void bst::insert(int item)

{

tree\_node \*p=new tree\_node;

tree\_node \*parent;

p->data=item;

p->left=NULL;

p->right=NULL;

parent=NULL;

if(root==NULL)

root=p;

else

{

tree\_node \*ptr;

ptr=root;

while(ptr!=NULL)

{

parent=ptr;

if(item>ptr->data)

ptr=ptr->right;

else

ptr=ptr->left;

}

if(item<parent->data)

parent->left=p;

else

parent->right=p;

}

}

void bst::inordertrav()

{

inorder(root);

}

void bst::inorder(tree\_node \*ptr)

{

if(ptr!=NULL)

{

inorder(ptr->left);

cout<<" "<<ptr->data<<" ";

inorder(ptr->right);

}

}

void bst::postordertrav()

{

postorder(root);

}

void bst::postorder(tree\_node \*ptr)

{

if(ptr!=NULL)

{

postorder(ptr->left);

postorder(ptr->right);

cout<<" "<<ptr->data<<" ";

}

}

void bst::preordertrav()

{

preorder(root);

}

void bst::preorder(tree\_node \*ptr)

{

if(ptr!=NULL)

{

cout<<" "<<ptr->data<<" ";

preorder(ptr->left);

preorder(ptr->right);

}

}

void main()

{

bst b;

b.insert(52);

b.insert(25);

b.insert(50);

b.insert(15);

b.insert(40);

b.insert(45);

b.insert(20);

cout<<"inorder"<<endl;

b.inordertrav();

cout<<endl<<"postorder"<<endl;

b.postordertrav();

cout<<endl<<"preorder"<<endl;

b.preordertrav();

getchar();

getchar();

}