package kdtree;

import java.util.\* ;

/\*\*

\*

\* @author Pradyot Patil

\*/

public class Kdtree {

public static void main(String[] args) {

// TODO code application logic here

new Tree() ;

}

public static class Tree

{

node root ;

static boolean b ;

public Tree()

{

Scanner sc=new Scanner(System.in);

System.out.print("Enter the no. of keys;");

int n=sc.nextInt() ;

System.out.print("Enter the key: ");

int x=sc.nextInt() ;

int y=sc.nextInt() ;

root=new node(x,y,null,null,0);

int i ;

for(i=1;i<n;i++)

{

System.out.print("Enter the key: ");

int x1=sc.nextInt() ;

int y1=sc.nextInt() ;

node d=new node(x1,y1,null,null,0);

insert(d) ;

display(root) ;

System.out.println("------------------------------------------");

}

System.out.print("Enter a key to search : ");

x=sc.nextInt() ;

y=sc.nextInt() ;

node nt=new node(x,y,null,null,0);

b=false ;

search\_single(nt,root);

if(!b)

{

System.out.print("key is not present!\n");

}

System.out.print("Enter a range to search : ");

x=sc.nextInt() ;

y=sc.nextInt() ;

node first=new node(x,y,null,null,0);

x=sc.nextInt() ;

y=sc.nextInt() ;

node second=new node(x,y,null,null,0);

search\_range(first,second,root) ;

}

public void insert(node d)

{

node find=root ;

node k=find ;

while(find!=null)

{

k=find ;

if(find.ht%2==0)

{

if(d.x>=find.x)

find=find.r ;

else

find=find.l ;

}

else

{

if(d.y>=find.y)

find=find.r ;

else

find=find.l ;

}

}

if(k.ht%2==0)

{

if(d.x>=k.x)

k.r=d ;

else

k.l=d ;

}

else

{

if(d.y>=k.y)

k.r=d ;

else

k.l=d ;

}

d.ht=k.ht+1 ;

}

public void display(node k)

{

System.out.println("("+k.x+","+k.y+",ht="+k.ht+")");

if(k.l!=null)

display(k.l);

if(k.r!=null)

display(k.r);

}

void search\_single(node k,node find)

{

if(k.x==find.x && k.y==find.y)

{

System.out.print("Key is present at height= "+find.ht+"\n");

b=true ;

return ;

}

if(find.l!=null)

search\_single(k,find.l);

if(find.r!=null)

search\_single(k,find.r);

}

public void search\_range(node first,node second,node find)

{

if(find.x>=first.x && find.x<=second.x && find.y>=first.y && find.y<=second.y)

{

System.out.print("found: ("+find.x+","+find.y+")\n");

b=true ;

}

if(find.l!=null)

search\_range(first,second,find.l);

if(find.r!=null)

search\_range(first,second,find.r);

}

}

public static class node{

int x ;

int y ;

node r ;

node l ;

int ht ;

public node(int x,int y,node r,node l,int ht)

{

this.x=x ;

this.y=y ;

this.r=r ;

this.l=l ;

this.ht=ht ;

}

}

}

**OUTPUT :**

Enter the no. of keys : 7

Enter the key: 3 6

Enter the key: 17 15

-----------------------------------------

(3,6,ht=0)

(17,15,ht=1)

------------------------------------------

Enter the key: 13 15

(3,6,ht=0)

(17,15,ht=1)

(13,15,ht=2)

------------------------------------------

Enter the key: 6 12

(3,6,ht=0)

(17,15,ht=1)

(6,12,ht=2)

(13,15,ht=2)

------------------------------------------

Enter the key: 9 1

(3,6,ht=0)

(17,15,ht=1)

(6,12,ht=2)

(9,1,ht=3)

(13,15,ht=2)

------------------------------------------

Enter the key: 2 7

(3,6,ht=0)

(2,7,ht=1)

(17,15,ht=1)

(6,12,ht=2)

(9,1,ht=3)

(13,15,ht=2)

------------------------------------------

Enter the key: 10 19

(3,6,ht=0)

(2,7,ht=1)

(17,15,ht=1)

(6,12,ht=2)

(9,1,ht=3)

(13,15,ht=2)

(10,19,ht=3)

------------------------------------------

Enter a key to search : 9 1

Key is present at height= 3

Enter a range to search : 1 1 , 12 12

found: (3,6)

found: (2,7)

found: (6,12)

found: (9,1)