CG Assignment 8

Yash Sarang D6AD/47

Aim:

Implenienting line clipping Algorithm Cohen Sutherland / Liang Barsky.

Theory:

Cohen Sutherland is one of the oldest and most popular line clipping algorithm. To speed up the process of this algorithm, it performs initial tests that reduces the number of intersections that must be calculated.

It does so by using a 4-bit code called as region code or outcodes. These codes ~~o~~ identify location of the end point of line.

Each bit position indicates a direction, starting from the rightmost position of each bit indicates left, right, bottom, top respectively.

Once we establish region codes for both the endpoints of a line we determined whether the endpoint is visible, partially visible or ~~inst~~ invisible with the help of ANDing of the region codes.

There are 3 cases which are explained in the ~~of~~ algorithm below in step 4

- Algorithm :-

① Read 2 end points of line as $p1\ (x_1, y_1)$ & $p2\ (x_2, y_2)$

② Read 2 corner points of the clipping ~~wind~~ window (left top & right both as (wx_1, wy_1) & (wx_2, wy_2)

③ Assign the region codes for 2 endpoints p1 & p2 using following steps:-

Initialize code with 0000.

Set bit 1 if $x < wx_1$.

Set bit 2 if $x > wx_2$.

Set bit 3 if $y < wy_1$.

Set bit 4 if $y > wy_2$.

④ Check for visibility of line

ⓐ If region codes for both endpoints are zero then line is completely visible. Draw the line & go to step 9.

ⓑ If region codes for endpoints are not zero & logical ANDing of them is also non zero then the line is invisible. Discard the line & move to step 9.

ⓒ If it does not satisfy 4ⓐ & 4ⓑ then the line is partially visible.

5) Determine the intersecting edge of clipping window as follows :-

ⓐ If region codes for both endpoints are nonzero find intersecting points p1 & p2 with boundary edges.

ⓑ If region codes for any one endpoint is nonzero then find intersection point p1 or P2.

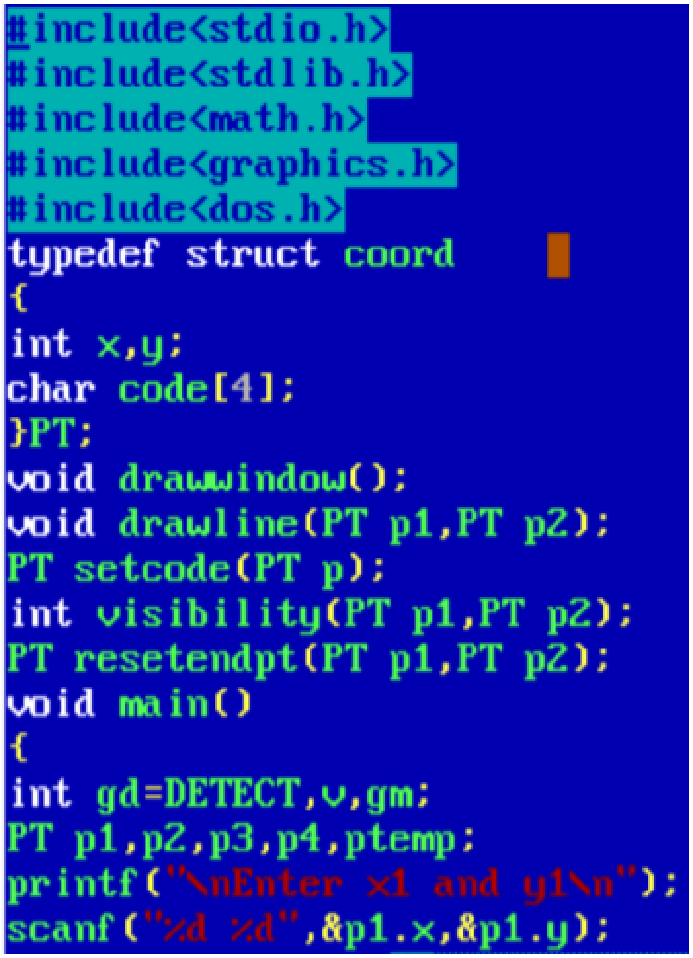
⑥ Divide the line segments considering intersection points.

7) Reject line segments if any endpoints of line appears outside of any boundary

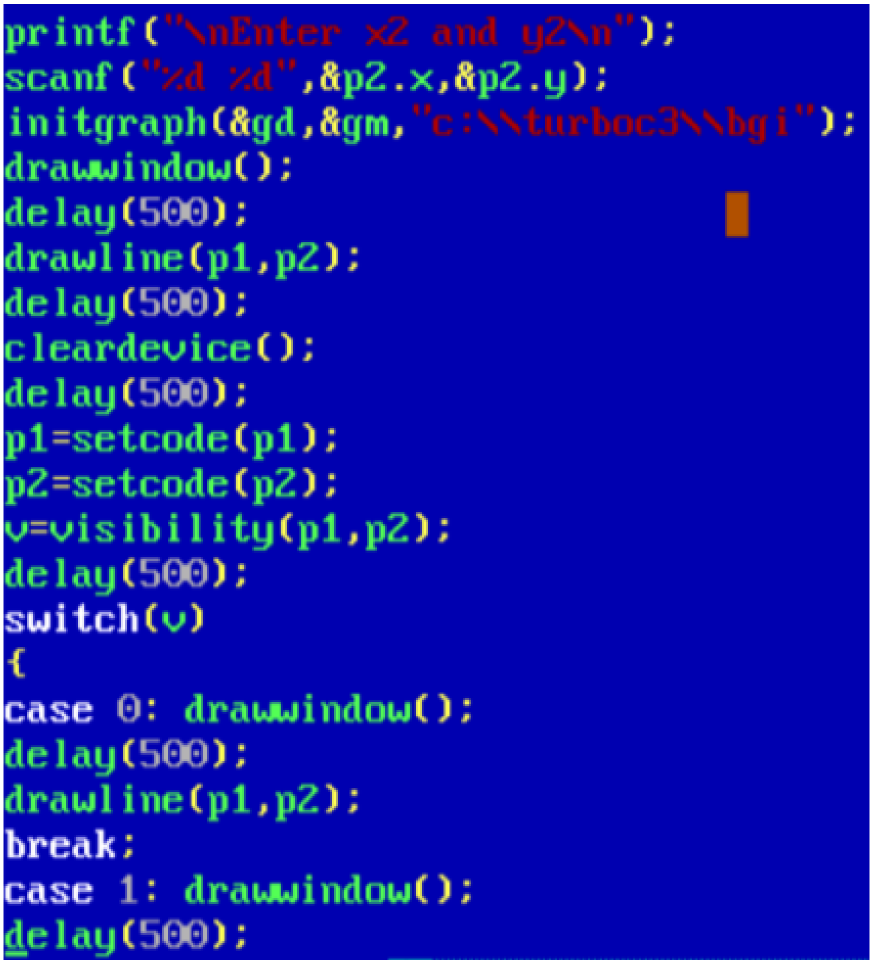
⑧ Draw the clipped line segment.

⑨ Stop.

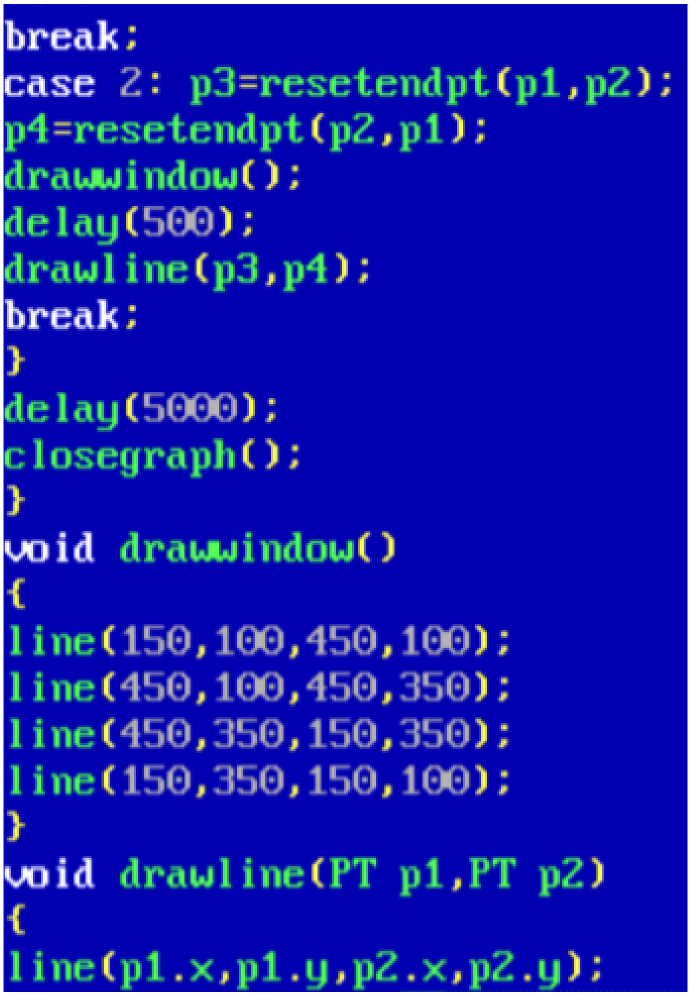
Code:-



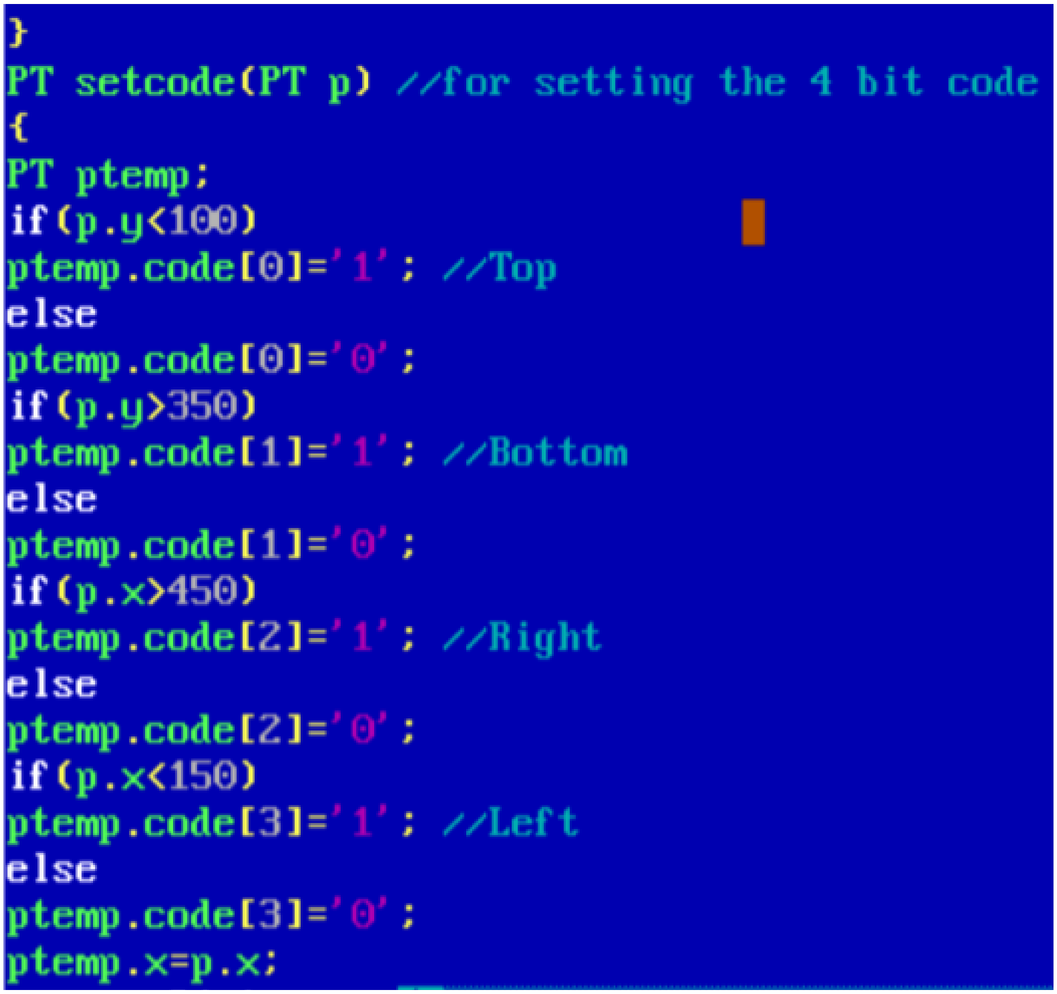
```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<graphics.h>
#include<dos.h>
typedef struct coord
{
int x,y;
char code[4];
}PT;
void drawwindow();
void drawline(PT p1,PT p2);
PT setcode(PT p);
int visibility(PT p1,PT p2);
PT resetendpt(PT p1,PT p2);
void main()
{
int gd=DETECT,v,gm;
PT p1,p2,p3,p4,ptemp;
printf("\nEnter x1 and y1\n");
scanf("%d %d",&p1.x,&p1.y);
```



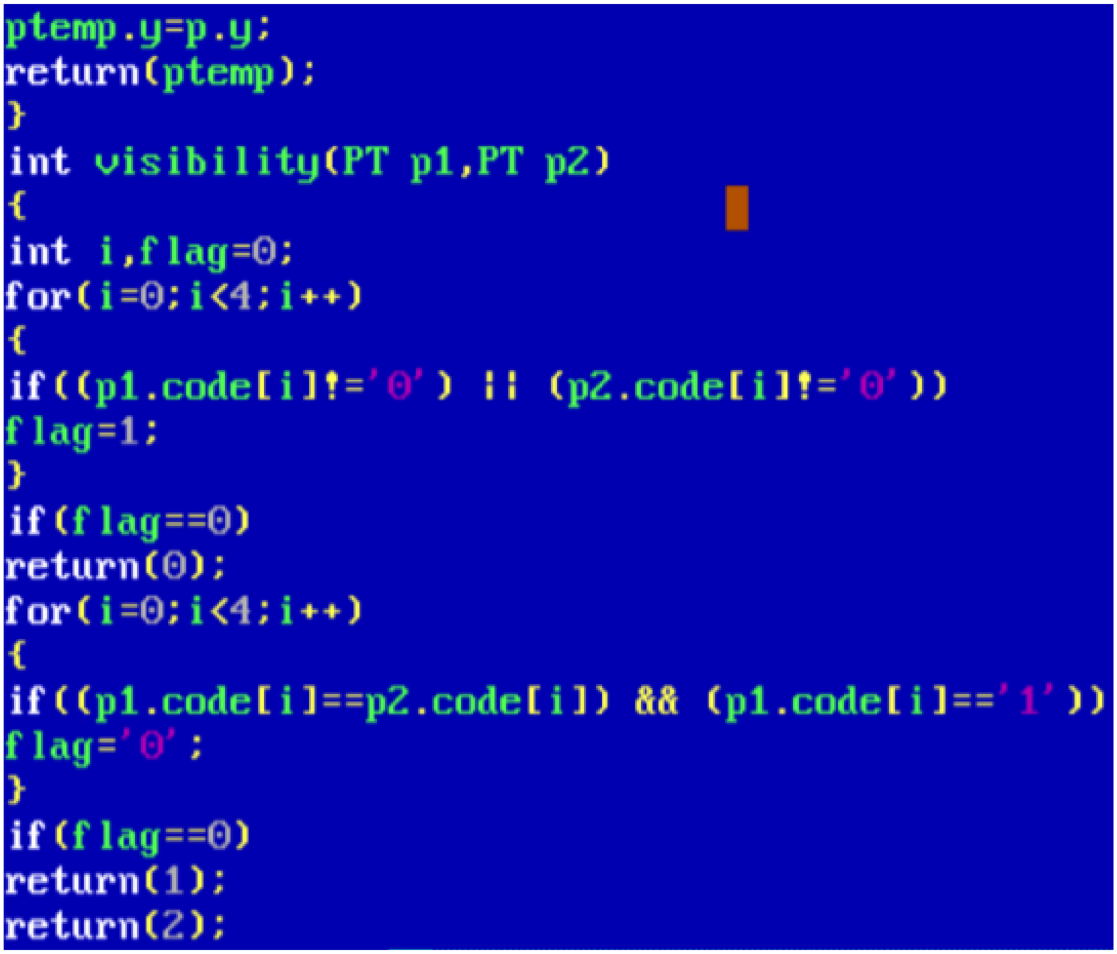
```
printf("\nEnter x2 and y2\n");
scanf("%d %d",&p2.x,&p2.y);
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
drawwindow();
delay(500);
drawline(p1,p2);
delay(500);
cleardevice();
delay(500);
p1=setcode(p1);
p2=setcode(p2);
v=visibility(p1,p2);
delay(500);
switch(v)
{
case 0: drawwindow();
delay(500);
drawline(p1,p2);
break;
case 1: drawwindow();
delay(500);
```



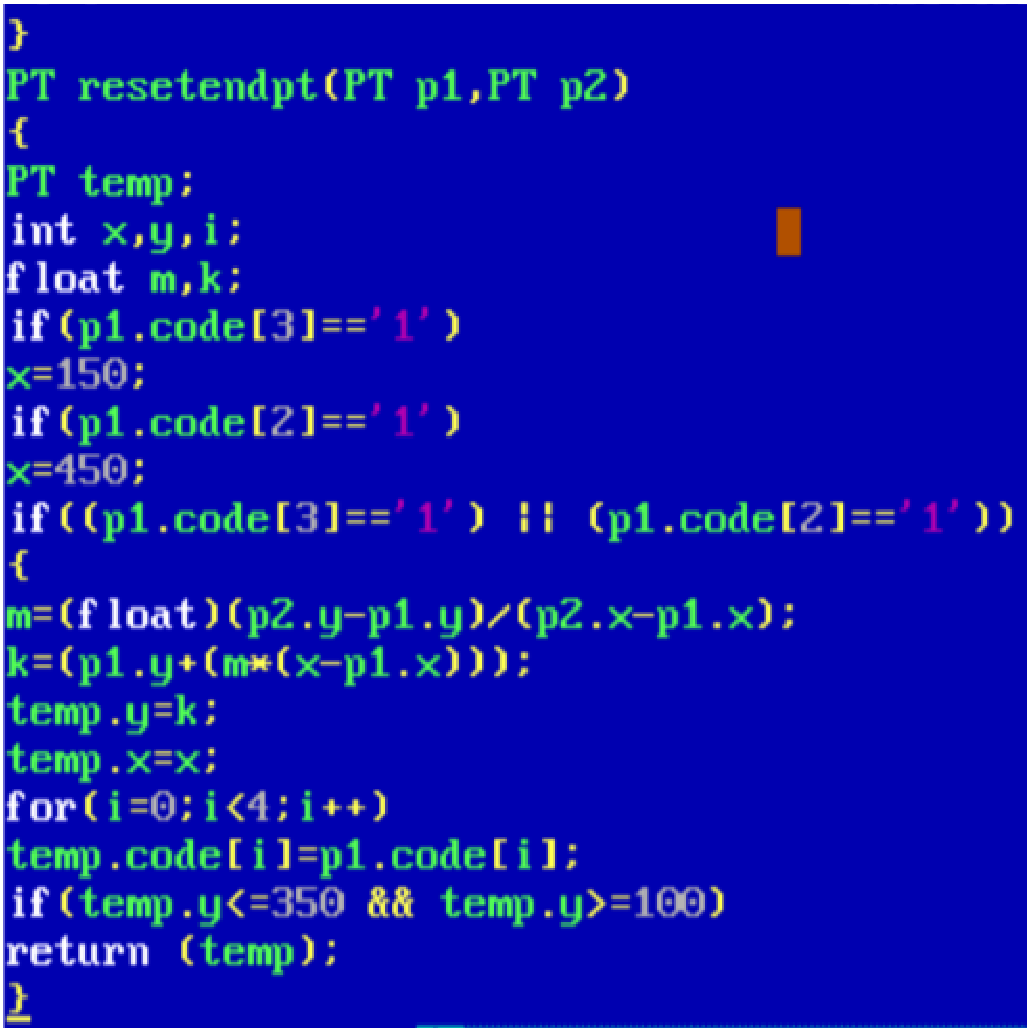
```
break;
case 2: p3=resetendpt(p1,p2);
p4=resetendpt(p2,p1);
drawwindow();
delay(500);
drawline(p3,p4);
break;
}
delay(5000);
closegraph();
}
void drawwindow()
{
line(150,100,450,100);
line(450,100,450,350);
line(450,350,150,350);
line(150,350,150,100);
}
void drawline(PT p1,PT p2)
{
line(p1.x,p1.y,p2.x,p2.y);
```



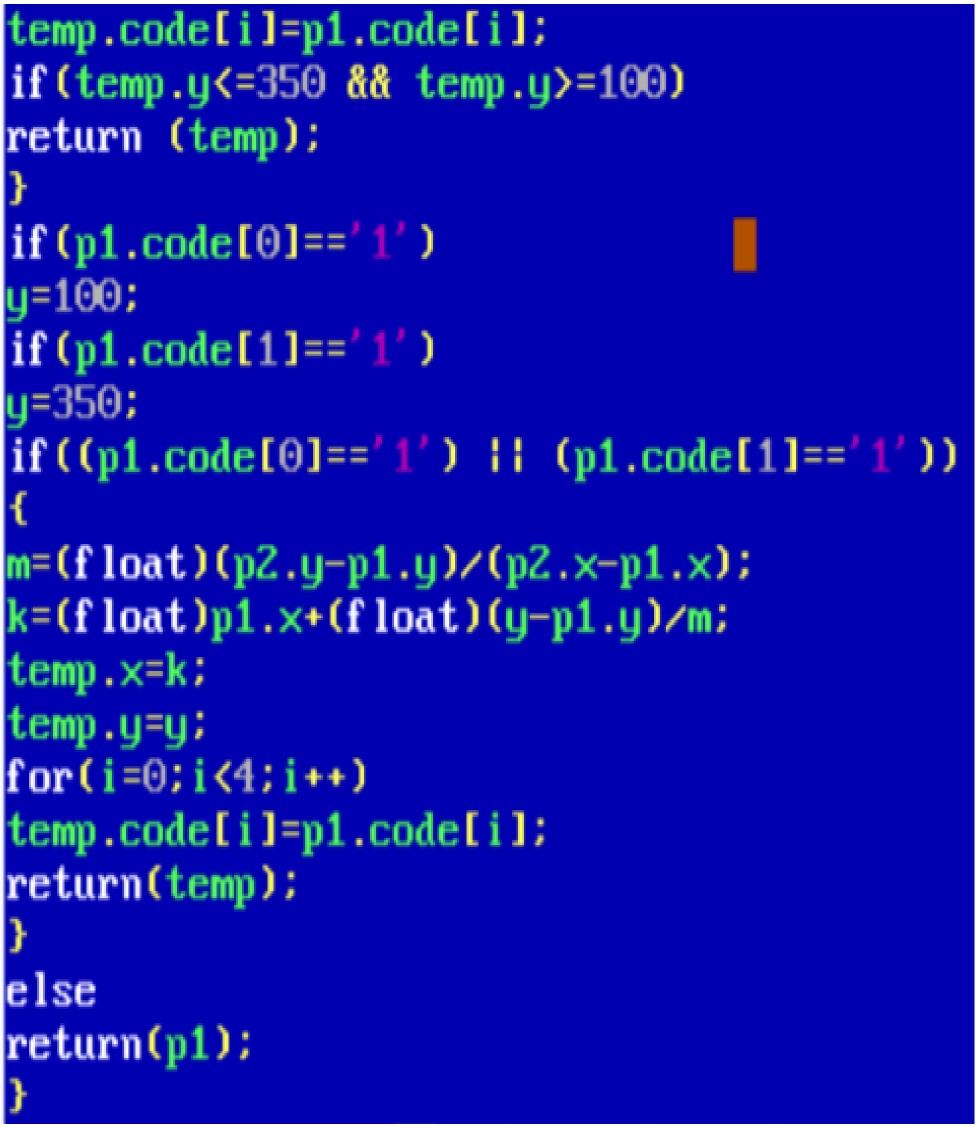
```
}
PT setcode(PT p) //for setting the 4 bit code
{
PT ptemp;
if(p.y<100)
ptemp.code[0]='1'; //Top
else
ptemp.code[0]='0';
if(p.y>350)
ptemp.code[1]='1'; //Bottom
else
ptemp.code[1]='0';
if(p.x>450)
ptemp.code[2]='1'; //Right
else
ptemp.code[2]='0';
if(p.x<150)
ptemp.code[3]='1'; //Left
else
ptemp.code[3]='0';
ptemp.x=p.x;
```



```
ptemp.y=p.y;
return(ptemp);
}
int visibility(PT p1,PT p2)
{
int i,flag=0;
for(i=0;i<4;i++)
{
if((p1.code[i]!='0') || (p2.code[i]!='0'))
flag=1;
}
if(flag==0)
return(0);
for(i=0;i<4;i++)
{
if((p1.code[i]==p2.code[i]) && (p1.code[i]=='1'))
flag='0';
}
if(flag==0)
return(1);
return(2);
```

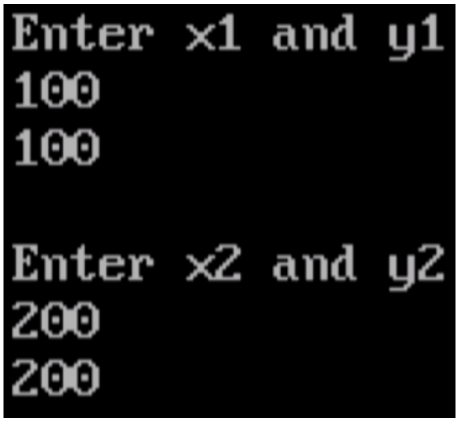


```
}
PT resetendpt(PT p1,PT p2)
{
PT temp;
int x,y,i;
float m,k;
if(p1.code[3]=='1')
x=150;
if(p1.code[2]=='1')
x=450;
if((p1.code[3]=='1') || (p1.code[2]=='1'))
{
m=(float)(p2.y-p1.y)/(p2.x-p1.x);
k=(p1.y+(m*(x-p1.x)));
temp.y=k;
temp.x=x;
for(i=0;i<4;i++)
temp.code[i]=p1.code[i];
if(temp.y<=350 && temp.y>=100)
return (temp);
}
```



```
temp.code[i]=p1.code[i];
if(temp.y<=350 && temp.y>=100)
return (temp);
}
if(p1.code[0]=='1')
y=100;
if(p1.code[1]=='1')
y=350;
if((p1.code[0]=='1') || (p1.code[1]=='1'))
{
m=(float)(p2.y-p1.y)/(p2.x-p1.x);
k=(float)p1.x+(float)(y-p1.y)/m;
temp.x=k;
temp.y=y;
for(i=0;i<4;i++)
temp.code[i]=p1.code[i];
return(temp);
}
else
return(p1);
}
```

Output :-



```
Enter x1 and y1
100
100

Enter x2 and y2
200
200
```