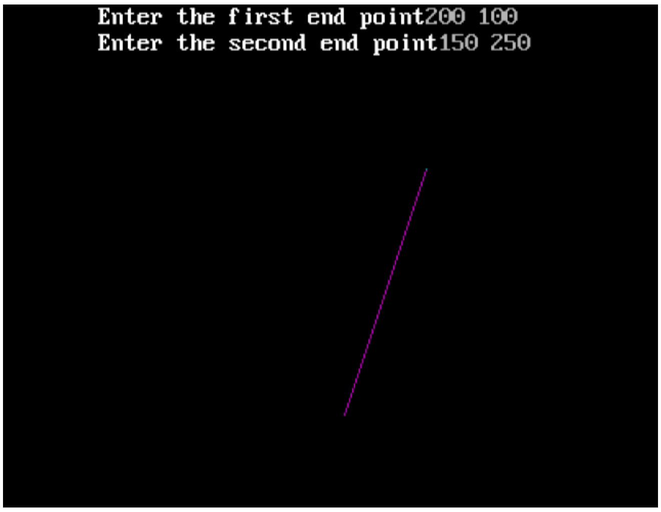
Problem Statement 1: Write a program to draw a line using DDA line generation algorithm.

Code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
int x1,x2,i,y1,y2,dx,dy,step;
float x,y,xin,yin;
int gm,gd=DETECT;
initgraph(&gd,&gm,"//turboc3//bgi");
printf("Enter the first end point");
scanf("%d%d",&x1,&y1);
printf("Enter the second end point");
scanf("%d%d",&x2,&y2);
dx=(x2-x1);
dy=(y2-y1);
if(abs(dx)>abs(dy))
step=abs(dx);
else
step=abs(dy);
xin=(float)dx/step;
yin=(float)dy/step;
x=x1,y=y1;
putpixel(x1,y1,3);
for(i=1;i<=step;i++)
{
x=float(x+xin);
y=float(y+yin);
putpixel(float(x+0.5),floor(y+0.5),5);
}
getch();
}
```

Output:



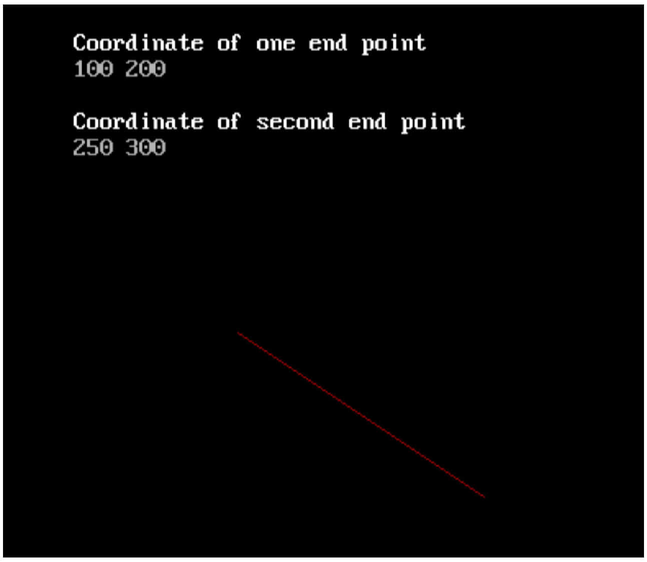
```
Enter the first end point200 100
Enter the second end point150 250
```

Problem Statement 2: Write a program to draw a line using Bresenham’s line generation algorithm.

Code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
int gd=DETECT,ystart,gm,p,dx,dy,x1,x2,y1,y2,xstart,xend;
initgraph(&gd,&gm,"//turboc3//bgi");
printf("\nCoordinate of one end point\n");
scanf("%d%d",&x1,&y1);
printf("\nCoordinate of second end point\n");
scanf("%d%d",&x2,&y2);
if(x1<x2)
{
xstart=x1;
xend=x2;
ystart=y1;
}
else
{
xstart=x2;
xend=x1;
ystart=y2;
}
putpixel(xstart,ystart,RED);
dx=abs(x1-x2);
dy=abs(y1-y2);
p=(2*dy-dx);
while(xstart<xend)
{
if(p<0)
{
xstart++;
p=(p+2*dy);
}
else
{
xstart++;
ystart++;
p=p+(dy-dx)*2;
}
putpixel(xstart,ystart,RED);
}
getch();
}
```

Output:



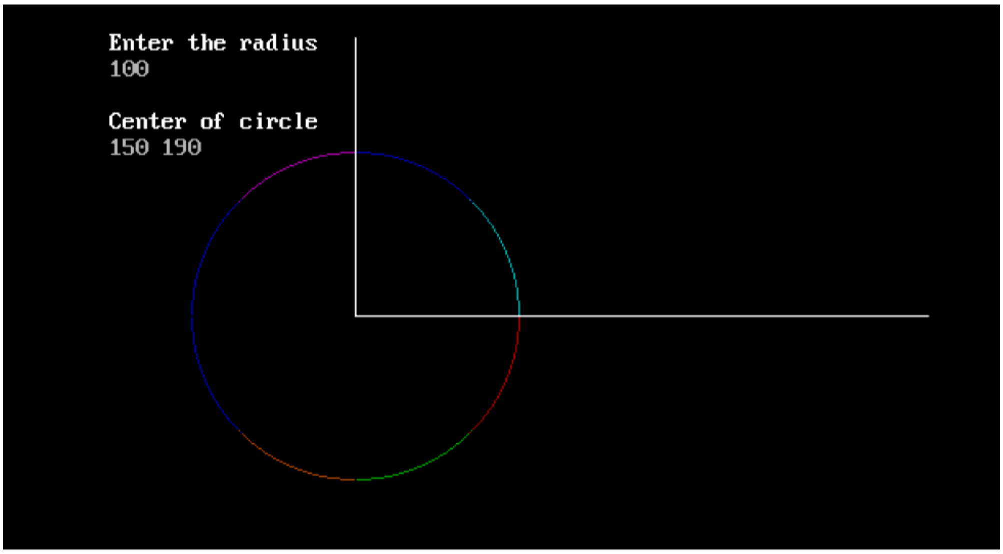
```
Coordinate of one end point
100 200

Coordinate of second end point
250 300
```

Problem Statement 3: Write a program to draw a circle using midpoint circle generation algorithm.

Code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm,r,x,y,xc,yc;
float p;
initgraph(&gd,&gm,"//turboc3//bgi");
printf("\nEnter the radius\n");
scanf("%d",&r);
printf("\nCenter of circle\n");
scanf("%d%d",&xc,&yc);
line(xc,yc,500,yc);
line(xc,yc,xc,20);
x=0;
y=r;
p=(5/4)-r;
while(x<=y)
{
if(p<0)
{
x=x+1;
p=2*x+p+1;
}
else
{
x=x+1;
y=y-1;
p=2*x+p+1-2*y;
}
putpixel(x+xc,y+yc,2);
putpixel(x+xc,yc-y,1);
putpixel(xc-x,y+yc,6);
putpixel(xc-x,yc-y,5);
putpixel(xc+y,yc+x,4);
putpixel(xc+y,yc-x,3);
putpixel(xc-y,yc+x,1);
putpixel(xc-y,yc-x,1);
}
getch();
}
```

Output:



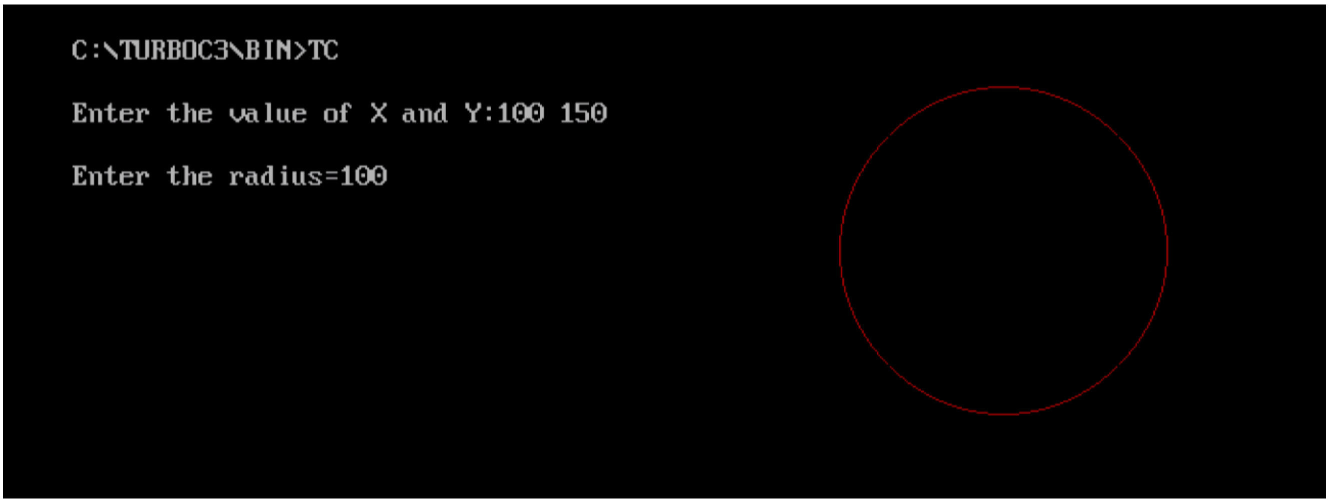
```
Enter the radius
100

Center of circle
150 190
```

Problem Statement 4: Write a program to draw a circle using Bresenham’s circle generating algorithm.

Code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void drawCircle(int xc,int yc,int x,int y)
{
putpixel(xc+x,yc+y,RED);
putpixel(xc-x,yc+y,RED);
putpixel(xc+x,yc-y,RED);
putpixel(xc-x,yc-y,RED);
putpixel(xc+y,yc+x,RED);
putpixel(xc-y,yc+x,RED);
putpixel(xc+y,yc-x,RED);
putpixel(xc-y,yc-x,RED);
}
void circleBres(int xc,int yc,int r)
{
int x=0,y=r;
int d=3-2*r;
drawCircle(xc,yc,x,y);
while(y>=x)
{
x++;
if(d>0)
{
y--;
d=d+4*(x-y)+10;
}
Else
d=d+4*x+6;
drawCircle(xc,yc,x,y);
delay(50);
}
}
int main()
{
int xc,yc,r;
printf("\nEnter the value of X and Y:");
scanf("%d%d",&xc,&yc);
printf("\nEnter the radius=");
scanf("%d",&r);
int gd=DETECT,gm;
initgraph(&gd,&gm,"//turboc3//bgi");
circleBres(xc,yc,r);
return 0;
}
```

Output:



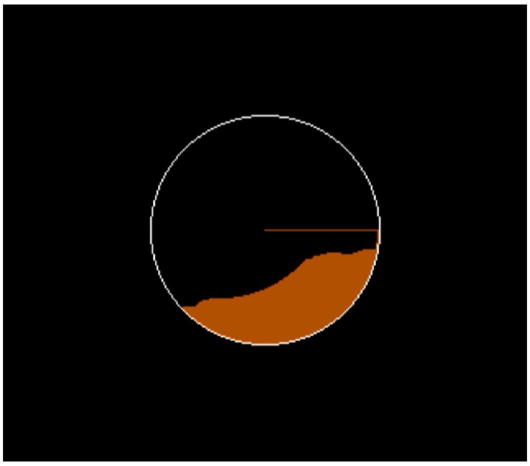
```
C:\TURBOC3\BIN>TC

Enter the value of X and Y:100 150

Enter the radius=100
```

Problem Statement 5: Write a program to implement boundary fill algorithm to fill a triangle.

Code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void boundaryFill4(int x,int y,int fill_color,int boundary_color)
{
if(getpixel(x,y)!=boundary_color && getpixel(x,y)!=fill_color)
{
putpixel(x,y,fill_color);
boundaryFill4(x+1,y,fill_color,boundary_color);
boundaryFill4(x,y+1,fill_color,boundary_color);
boundaryFill4(x-1,y,fill_color,boundary_color);
boundaryFill4(x,y-1,fill_color,boundary_color);
}
}
int main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"//turboc3//bgi");
int x=250,y=200,radius=70;
circle(x,y,radius);
boundaryFill4(x,y,6,15);
getch();
delay(10000);
closegraph();
return 0;
}
```

Output:

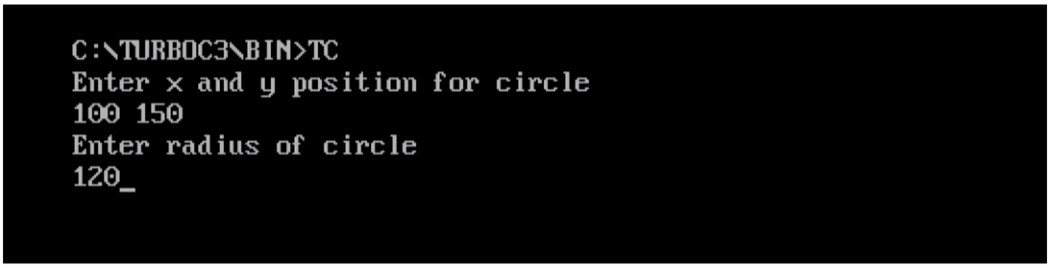


Problem Statement 6: Write a program to implement flood fill algorithm to fill a circle.

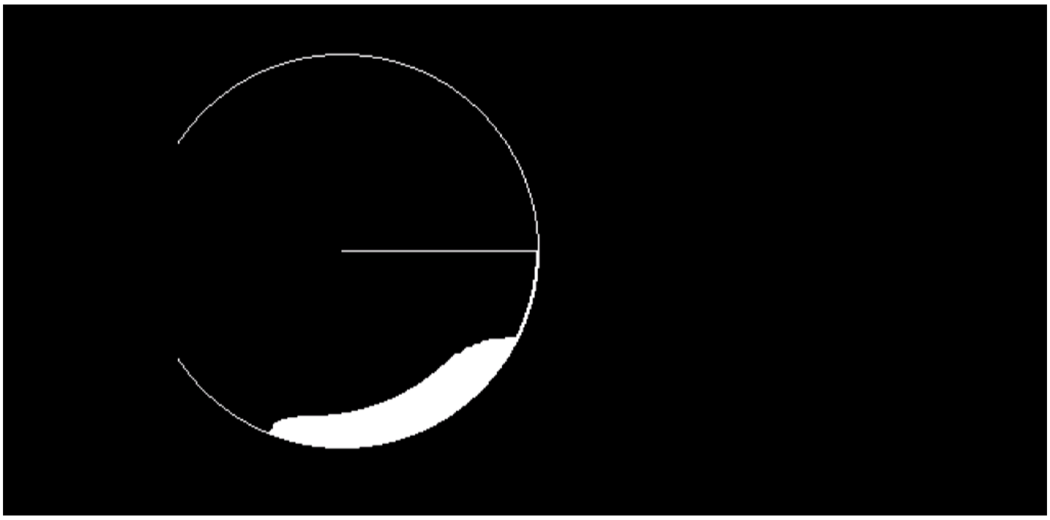
Code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void floodFill(int x,int y,int oldcolor,int newcolor)
{
if(getpixel(x,y)==oldcolor)
{
putpixel(x,y,newcolor);
floodFill(x+1,y,oldcolor,newcolor);
floodFill(x,y+1,oldcolor,newcolor);
floodFill(x-1,y,oldcolor,newcolor);
floodFill(x,y-1,oldcolor,newcolor);
}
}
int main()
{
int gm,gd=DETECT,radius;
int x,y;
printf("Enter x and y position for circle\n");
scanf("%d%d",&x,&y);
printf("Enter radius of circle\n");
scanf("%d",&radius);
initgraph(&gd,&gm,"//turboc3 //bgi");
circle(x,y,radius);
floodFill(x,y,0,15);
delay(5000);
closegraph();
return 0;
}
```

Output:



```
C:\TURBOC3\BIN>TC
Enter x and y position for circle
100 150
Enter radius of circle
120_
```



Problem Statement 7: Write a program to implement the Liang-Barsky Line clipping algorithm.

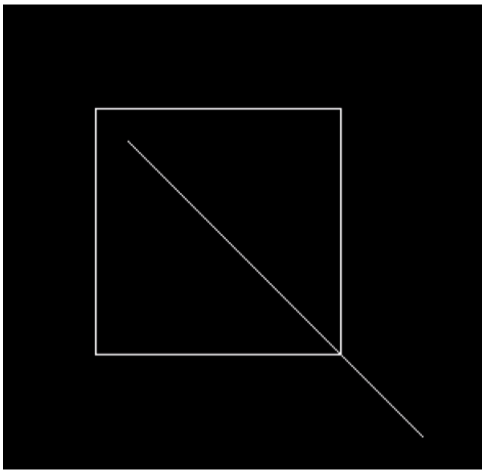
Code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
void main()
{
int i,gd=DETECT,gm;
int x1,x2,y1,y2,xmin,ymin,xmax,ymax,xx1,xx2,yy1,yy2,dx,dy;
float t1,t2,p[4],q[4],temp;
x1=120;
y1=120;
x2=300;
y2=300;
xmin=100;
ymin=100;
xmax=250;
ymax=250;
initgraph(&gd,&gm,"\\turboc3\\bgi");
rectangle(xmin,ymin,xmax,ymax);
dx=x2-x1;
dy=y2-y1;
p[0]=-dx;
p[1]=dx;
p[2]=-dy;
p[3]=dy;
q[0]=x1-xmin;
q[1]=xmax-x1;
q[2]=y1-ymin;
```

```
q[3]=ymax-y1;
for(i=0;i<4;i++)
{
if(p[i]==0)
{
printf("Line is parallel to one of the clipping window");
if(q[i]>0)
{
if(i<2)
{
if(y1<ymin)
{
y2=ymax;
}
line(x1,y1,x2,y2);
}
if(i>1)
{
if(x1<xmin)
{
x1=xmin;
}
if(x2>xmax)
{
x2=xmax;
}
line(x1,y1,x2,y2);
}
}
}
}
```

```
t1=0;
t2=1;
for(i=0;i<4;i++)
{
temp=q[i]/p[i];
if(p[i]<0)
{
if(t1<=temp)
t1=temp;
}
else
{
if(t2<=temp)
t2=temp;
}
}
if(t1<t2)
{
xx1=x1+t1*p[1];
xx2=x1+t2*p[1];
yy1=y1+t1*p[3];
yy2=y1+t2*p[3];
line(xx1,yy1,xx2,yy2);
}
delay(5000);
closegraph();
}
```

Output:

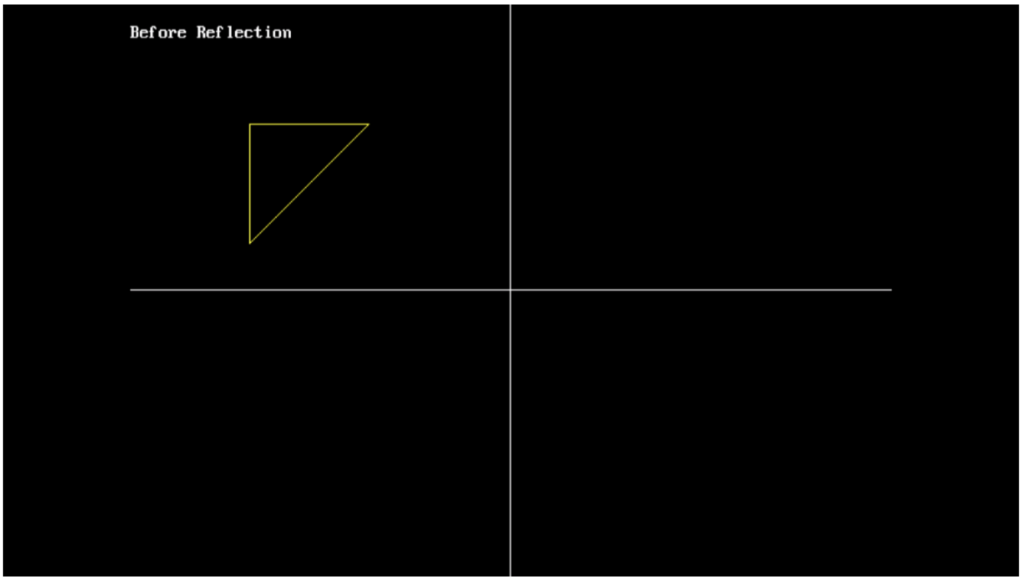


Problem Statement 8: Write a program to implement 2D reflection of a triangle.

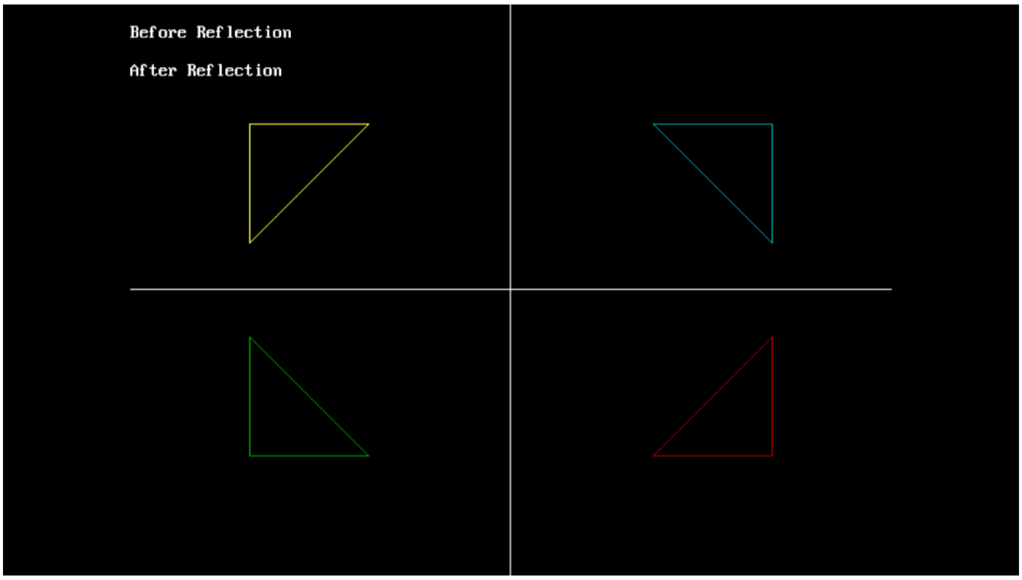
Code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
{
int gm,gd=DETECT,ax,x1=100;
int x2=100,x3=200,y1=100;
int y2=200,y3=100;
initgraph(&gd,&gm,"//turboc3//bgi");
cleardevice();
line(getmaxx()/2,0,getmaxx()/2,getmaxy());
line(0,getmaxy()/2,getmaxx(),getmaxy()/2);
printf("\nBefore Reflection\n");
setcolor(14);
line(x1,y1,x2,y2);
line(x2,y2,x3,y3);
line(x3,y3,x1,y1);
getch();
printf("\nAfter Reflection\n");
setcolor(4);
line(getmaxx()-x1,getmaxy()-y1,getmaxx()-x2,getmaxy()-y2);
line(getmaxx()-x2,getmaxy()-y2,getmaxx()-x3,getmaxy()-y3);
line(getmaxx()-x3,getmaxy()-y3,getmaxx()-x1,getmaxy()-y1);
setcolor(3);
line(getmaxx()-x1,y1,getmaxx()-x2,y2);
line(getmaxx()-x2,y2,getmaxx()-x3,y3);
line(getmaxx()-x3,y3,getmaxx()-x1,y1);
setcolor(2);
line(x1,getmaxy()-y1,x2,getmaxy()-y2);

line(x2,getmaxy()-y2,x3,getmaxy()-y3);

line(x3,getmaxy()-y3,x1,getmaxy()-y1);

getch();

closegraph();

}
```

Output:



```
Before Reflection
```



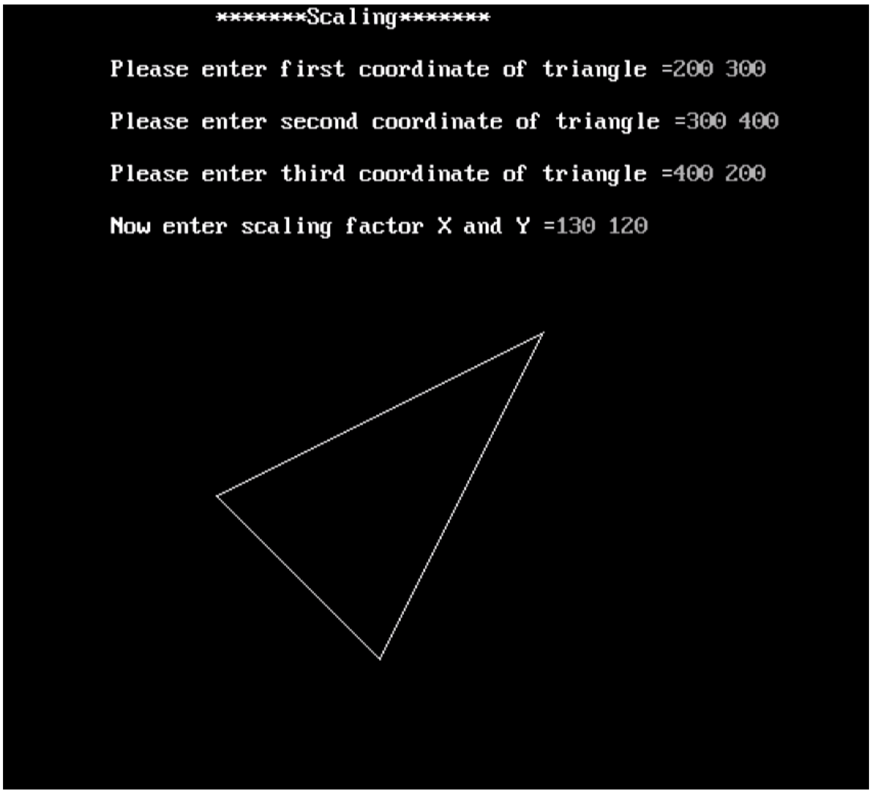
```
Before Reflection
After Reflection
```

Problem Statement 9: Write a program to scale a triangle about origin.

Code:

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int x,y,x1,y1,x2,y2;
int scl_fctr_x,scl_fctr_y;
int gd=DETECT,gm;
initgraph(&gd,&gm,"//turboc3//bgi");
printf("\t\t\t *******Scaling*******\n");
printf("\n\t\t Please enter first coordinate of triangle =");
scanf("%d%d",&x,&y);
printf("\n\t\t Please enter second coordinate of triangle =");
scanf("%d%d",&x1,&y1);
printf("\n\t\t Please enter third coordinate of triangle =");
scanf("%d%d",&x2,&y2);
line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x,y);
printf("\n\t\t Now enter scaling factor X and Y =");
scanf("%d%d",&scl_fctr_x,&scl_fctr_y);
x=x*scl_fctr_x;
x1=x1*scl_fctr_x;
x2=x2*scl_fctr_x;
y=y*scl_fctr_y;
y1=y1*scl_fctr_y;
y2=y2*scl_fctr_y;
line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x,y);

getch();

closegraph();

}
```

Output:



```
*******Scaling*******

Please enter first coordinate of triangle =200 300

Please enter second coordinate of triangle =300 400

Please enter third coordinate of triangle =400 200

Now enter scaling factor X and Y =130 120
```

Problem Statement 10:

Code:

Output:

Problem Statement 11:

Code:

Output:

Problem Statement 12:

Code:

Output:

Problem Statement 13:

Code:

Output:

Problem Statement 13:

Code:

Output:

Problem Statement 14:

Code:

Output:

Problem Statement 15:

Code:

Output:

Problem Statement 16:

Code:

Output: