Parshvanath Charitable Trust's
A. P. SHAH INSTITUTE OF TECHNOLOGY
(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)
(Religious Jain Minority)

MoodleId:21102010 **Name: Kartik Kanchan**

Div: A **Department: Computer Engineering**

Batch:A3 **RollNo: 58**

Assigment no : 2

Problem Statement: Implement the following 2D transformations in C:

1. Reflection 2.

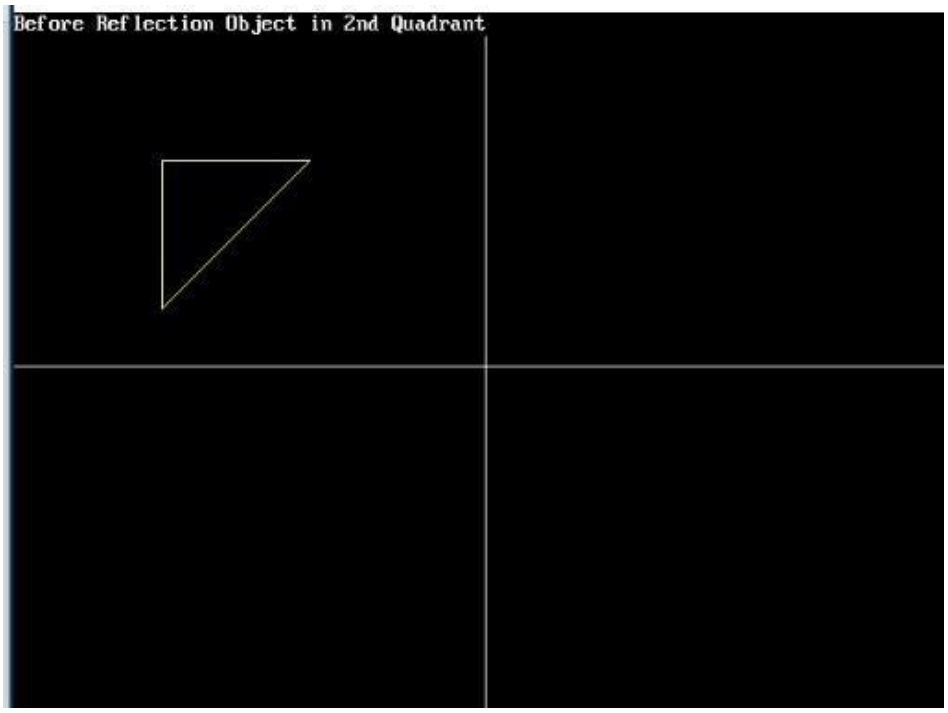
Shear (X and Y)

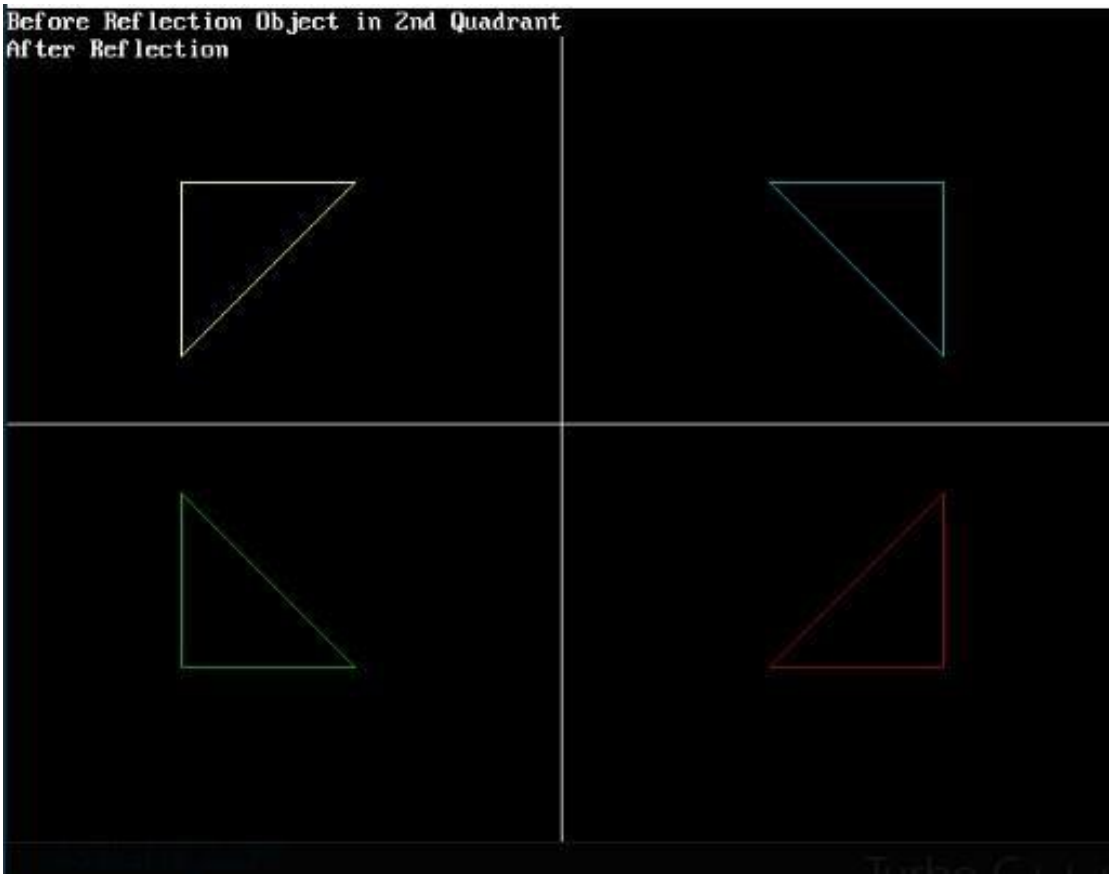
Program:

```
// C program for the above approach

#include <conio.h>
#include <graphics.h>
#include <stdio.h>

// Driver Code void
main()
{
      // Initialize the drivers    int gm,
gd = DETECT, ax, x1 = 100;   int x2  =
100, x3 = 200, y1 = 100;     int y2 =
200, y3 = 100;

        // Add in your BGI folder path
      // like below initgraph(&gd, &gm,
// "C:\\TURBOC3\\BGI");
        initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
        cleardevice();

        // Draw the graph
      line(getmaxx() / 2, 0, getmaxx() / 2,getmaxy());
line(0, getmaxy() / 2, getmaxx(),getmaxy() / 2);

        // Object initially at 2nd quadrant
        printf("Before Reflection Object"
               " in 2nd Quadrant");

      // Set the color
setcolor(14);    line(x1, y1,
x2, y2);    line(x2, y2, x3,
y3);        line(x3, y3, x1,
y1);        getch();
```

```
      // After reflection
printf("\nAfter Reflection");

        // Reflection along origin i.e.,
        // in 4th quadrant
        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);
        // Reflection along x-axis i.e.,
        // in 1st quadrant
        setcolor(3);
      line(getmaxx() - x1, y1,getmaxx() - x2, y2);
line(getmaxx() - x2, y2,getmaxx() - x3, y3);
line(getmaxx() - x3, y3,getmaxx() - x1, y1);

        // Reflection along y-axis i.e.,
        // in 3rd quadrant
        setcolor(2);
      line(x1, getmaxy() - y1, x2,getmaxy() - y2);
line(x2, getmaxy() - y2, x3,getmaxy() - y3);
line(x3, getmaxy() - y3, x1,getmaxy() - y1);
        getch();

        // Close the graphics
        closegraph();
}
```

Output:



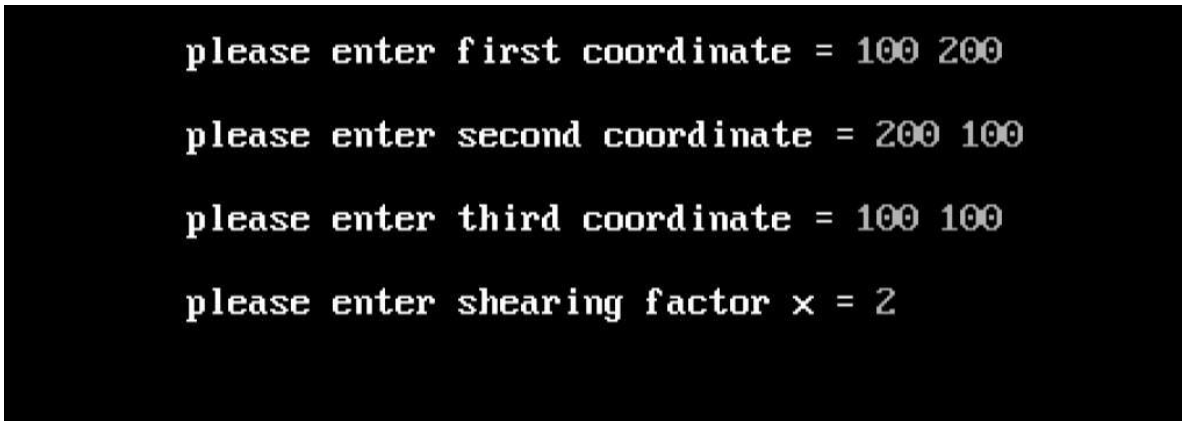


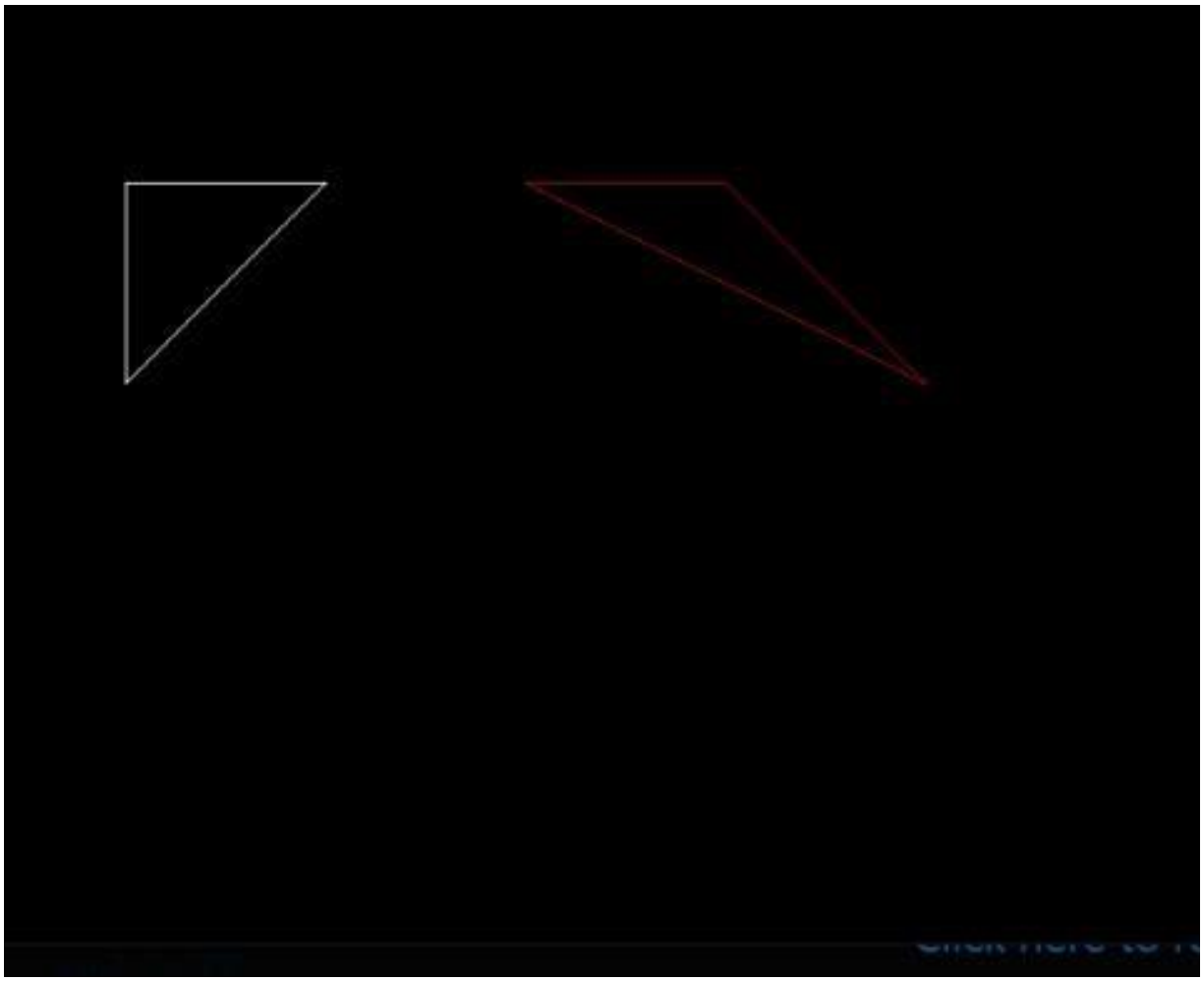
Program:

```
#include<stdio.h>
```

```
#include<graphics.h>
#include<conio.h>
void main() {
int gd=DETECT,gm; int x,y,x1,y1,x2,y2,shear_f;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
printf("\n please enter first coordinate = ");
scanf("%d %d",&x,&y);
printf("\n please enter second coordinate = ");
scanf("%d %d",&x1,&y1);
printf("\n please enter third coordinate = "); scanf("%d
%d",&x2,&y2);
printf("\n please enter shearing factor x = ");
scanf("%d",&shear_f); cleardevice();
line(x,y,x1,y1); line(x1,y1,x2,y2);
line(x2,y2,x,y);
 setcolor(RED);
x=x+ y*shear_f;
x1=x1+
y1*shear_f;
x2=x2+
y2*shear_f;
line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x,y);
getch();
closegraph(); }
```

Output:



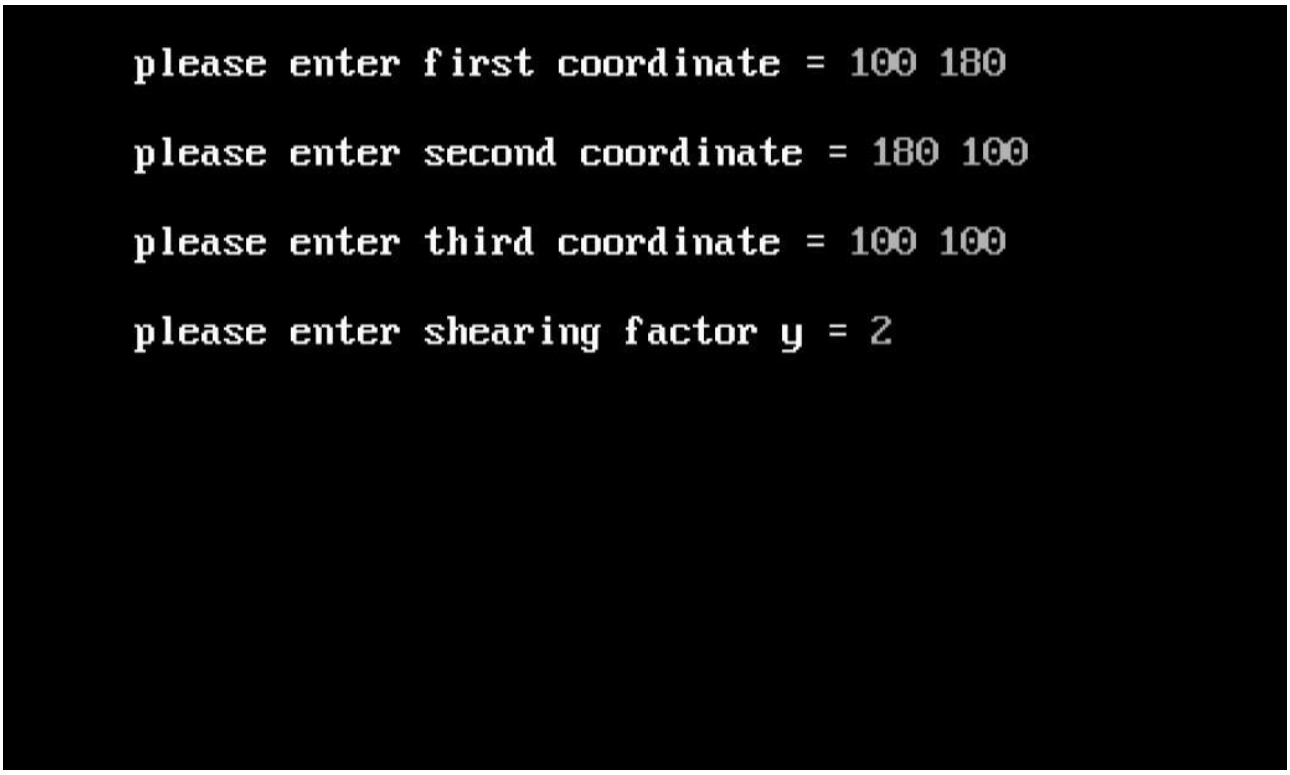


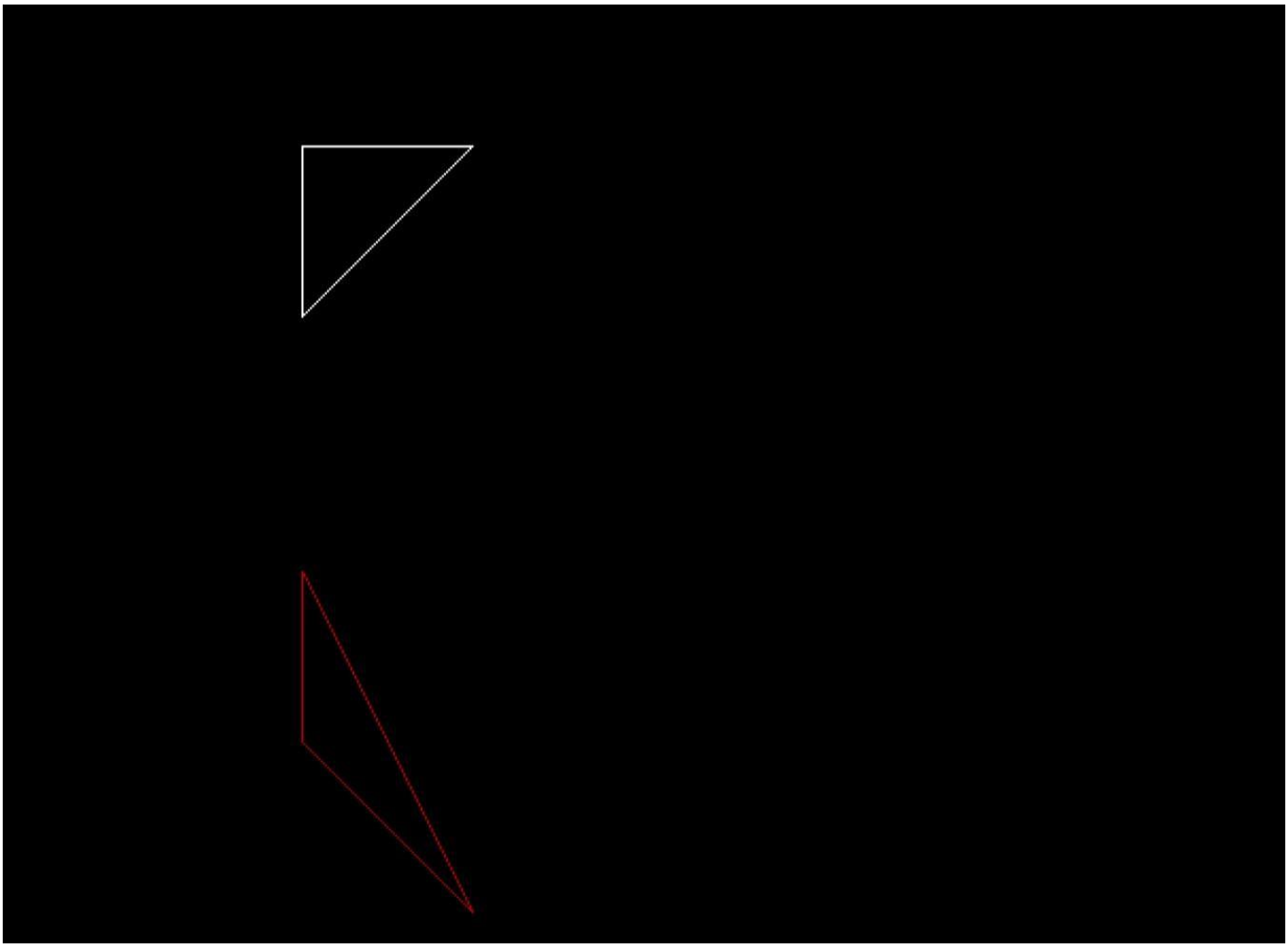
Program:

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h> void
main()
{
int gd=DETECT,gm; int x,y,x1,y1,x2,y2,shear_f;
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
printf("\n please enter first coordinate = ");
scanf("%d %d",&x,&y);
printf("\n please enter second coordinate = ");
scanf("%d %d",&x1,&y1);
printf("\n please enter third coordinate = "); scanf("%d
%d",&x2,&y2);
printf("\n please enter shearing factor x = ");
scanf("%d",&shear_f); cleardevice();
line(x,y,x1,y1);
line(x1,y1,x2,y2); line(x2,y2,x,y);
 setcolor(RED);
x=x+ y*shear_f;
```

```
x1=x1+ y1*shear_f;
x2=x2+ y2*shear_f;
line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x,y);
getch();
closegraph(); }
```

Output:





Problem Statement Implement Sutherland Hodgeman Polygon clipping algorithm in C

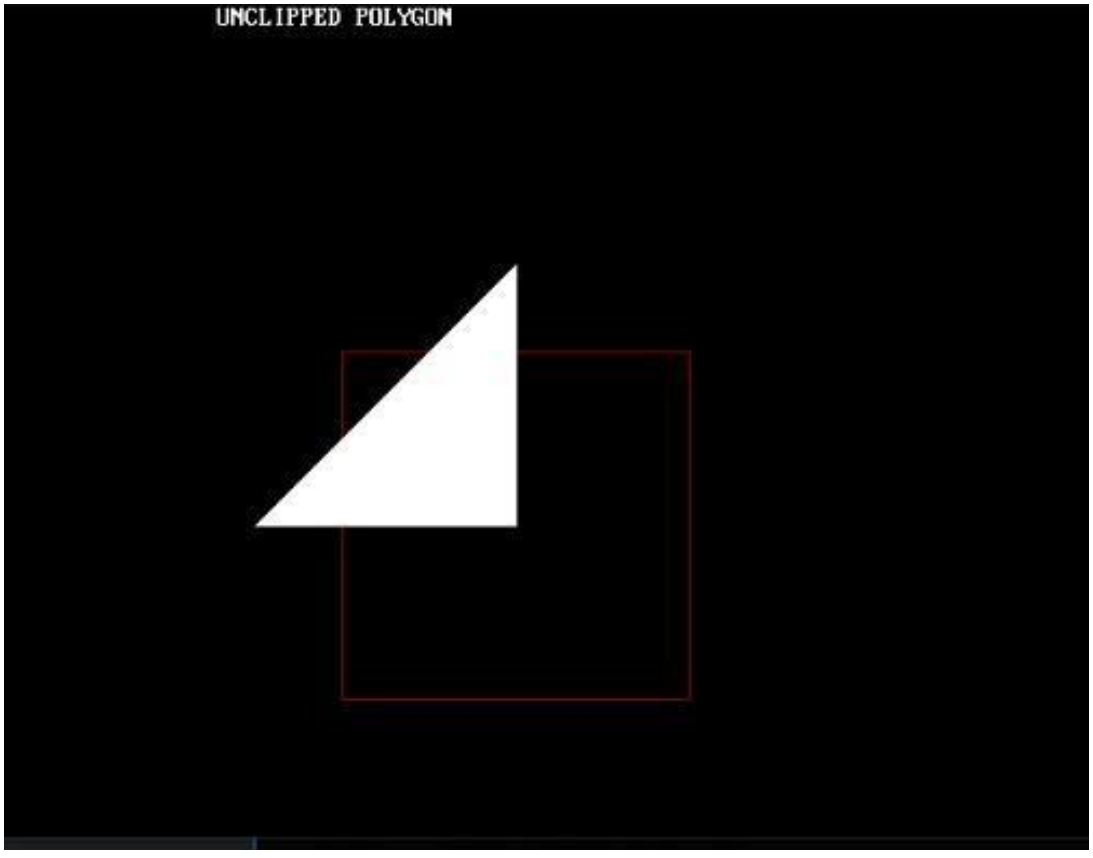
Program:

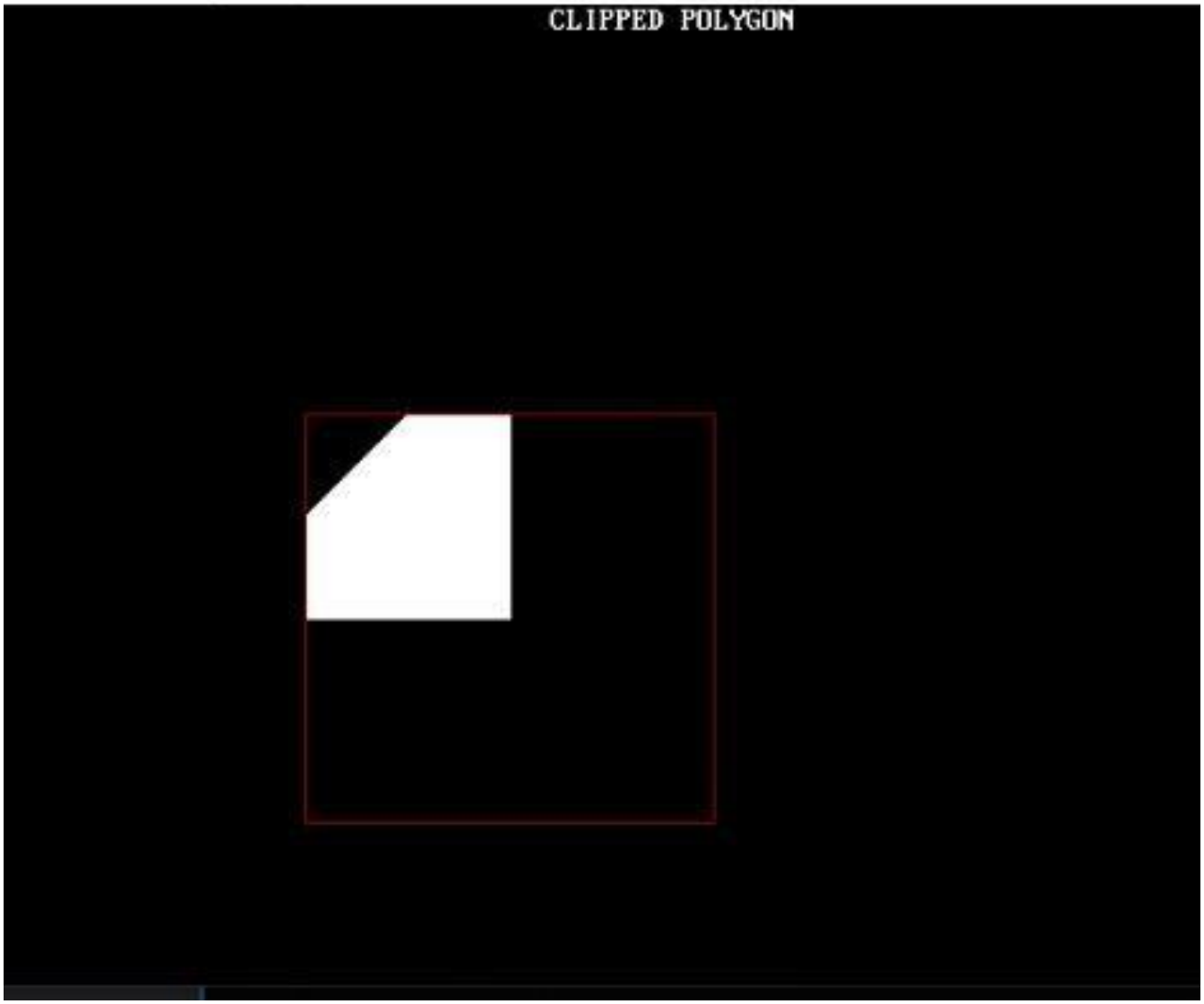
```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#define round(a) ((int)(a+0.5)) int
k;
float xmin,ymin,xmax,ymax,arr[20],m;
void clipl(float x1,float y1,float x2,float y2)
{     if(x2-
x1)
        m=(y2-y1)/(x2-x1);
else         m=100000;
    if(x1 >= xmin && x2 >= xmin)
    {
arr[k]=x2;
arr[k+1]=y2;
k+=2;     }
    if(x1 < xmin && x2 >= xmin)
    {
        arr[k]=xmin;
arr[k+1]=y1+m*(xmin-x1);
arr[k+2]=x2;
arr[k+3]=y2;          k+=4;     }
    if(x1 >= xmin  && x2 < xmin)
    {
        arr[k]=xmin;
arr[k+1]=y1+m*(xmin-x1);          k+=2;
    }
}
void clipt(float x1,float y1,float x2,float y2)
{     if(y2-
y1)
        m=(x2-x1)/(y2-y1);
else         m=100000;
    if(y1 <= ymax && y2 <= ymax)
    {
        arr[k]=x2;
arr[k+1]=y2;          k+=2;
}     if(y1 > ymax && y2 <=
ymax)
    {
        arr[k]=x1+m*(ymax-y1);
arr[k+1]=ymax;
arr[k+2]=x2;
arr[k+3]=y2;          k+=4;
}
    if(y1 <= ymax  && y2 > ymax)
    {
        arr[k]=x1+m*(ymax-y1);
arr[k+1]=ymax;          k+=2;
    }
}
void clipr(float x1,float y1,float x2,float y2)
{     if(x2-x1)          m=(y2-
y1)/(x2-x1);     else
m=100000;     if(x1 <= xmax &&
x2 <= xmax)
    {
arr[k]=x2;
arr[k+1]=y2;
k+=2;     }
```

```
    if(x1 > xmax && x2 <= xmax)
    {         arr[k]=xmax;
arr[k+1]=y1+m*(xmax-x1);
arr[k+2]=x2;
arr[k+3]=y2;          k+=4;     }
    if(x1 <= xmax  && x2 > xmax)
    {         arr[k]=xmax;
arr[k+1]=y1+m*(xmax-x1);
k+=2;
    }
}
void clipb(float x1,float y1,float x2,float y2)
{     if(y2-y1)         m=(x2-
x1)/(y2-y1);     else
m=100000;     if(y1 >= ymin &&
y2 >= ymin)
    {         arr[k]=x2;
arr[k+1]=y2;          k+=2;
}     if(y1 < ymin && y2 >=
ymin)
    {
        arr[k]=x1+m*(ymin-y1);
arr[k+1]=ymin;
arr[k+2]=x2;
arr[k+3]=y2;          k+=4;
}
    if(y1 >= ymin  && y2 < ymin)
    {
        arr[k]=x1+m*(ymin-y1);
arr[k+1]=ymin;          k+=2;
    }
}   void main() {     int
gd=DETECT,gm,n,poly[20];
float xi,yi,xf,yf,polyy[20];
clrscr();
    cout<<"Coordinates of rectangular clip window :\nxmin,ymin             :";
cin>>xmin>>ymin;
    cout<<"xmax,ymax             :";
cin>>xmax>>ymax;
    cout<<"\n\nPolygon to be clipped :\nNumber of sides       :";
cin>>n;
    cout<<"Enter the coordinates :";
for(int i=0;i < 2*n;i++)cin>>polyy[i];
polyy[i]=polyy[0];     polyy[i+1]=polyy[1];
    for(i=0;i < 2*n+2;i++)poly[i]=round(polyy[i]);
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
setcolor(RED);
    rectangle(xmin,ymax,xmax,ymin);
cout<<"\t\tUNCLIPPED POLYGON";
setcolor(WHITE);     fillpoly(n,poly);
       getch();
    cleardevice();
k=0;
    for(i=0;i < 2*n;i+=2)
           clipl(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);
n=k/2;     for(i=0;i < k;i++)
              polyy[i]=arr[i];
    polyy[i]=polyy[0];
polyy[i+1]=polyy[1];     k=0;
    for(i=0;i < 2*n;i+=2)
```

```
            clipt(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);
n=k/2;      for(i=0;i < k;i++)
               polyy[i]=arr[i];
    polyy[i]=polyy[0];
polyy[i+1]=polyy[1];      k=0;
    for(i=0;i < 2*n;i+=2)
clipr(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);      n=k/2;
for(i=0;i < k;i++)
            polyy[i]=arr[i];
polyy[i]=polyy[0];
polyy[i+1]=polyy[1];      k=0;
    for(i=0;i < 2*n;i+=2)
clipb(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);      for(i=0;i < k;i++)
               poly[i]=round(arr[i]);
    if(k)
               fillpoly(k/2,poly);
    setcolor(RED);
    rectangle(xmin,ymax,xmax,ymin);
cout<<"\tCLIPPED POLYGON";
getch();      closegraph(); }
```

Output:




CLIPPED POLYGON