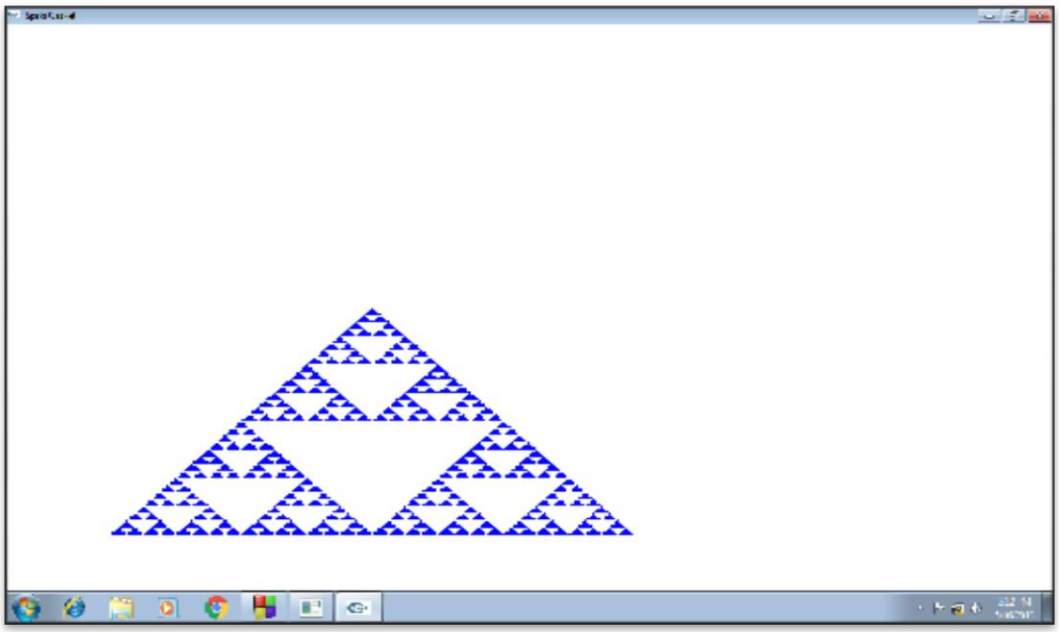
1.Drawing the basic primitives and sierpinsky gasket using openGL*.

```
#include<GL/gl.h>
#include<GL/glut.h>
#include<stdio.h>
#include<math.h>
void myInit()
{
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(0.0,10.0,0.0,10.0);
   glMatrixMode(GL_MODELVIEW);
   glClearColor(1.0,1.0,1.0,1.0);
   glColor3f(0.0,0.0,1.0);
}
void triangle(GLfloat *a,GLfloat *b,GLfloat *c)
{
   glVertex2fv(a);
   glVertex2fv(b);
   glVertex2fv(c);

}
void draw_triangle(GLfloat *a,GLfloat *b,GLfloat *c,int k)
{

   GLfloat ab[2],bc[2],ac[2];
   int j;
   if(k>1)
   {
      for(j=0;j<2;j++)
         ab[j]=(a[j]+b[j])/2.0;
      for(j=0;j<2;j++)
         bc[j]=(b[j]+c[j])/2.0;
      for(j=0;j<2;j++)
         ac[j]=(a[j]+c[j])/2.0;

      draw_triangle(a,ab,ac,k-1);
      draw_triangle(b,bc,ab,k-1);
      draw_triangle(c,ac,bc,k-1);
   }
   else
   {
```

```
      triangle(a,b,c);
   }
}

void display()
{
   GLfloat a[2]={1.0,1.0};
   GLfloat b[2]={6.0,1.0};
   GLfloat c[2]={3.5,5.0};
   glClear(GL_COLOR_BUFFER_BIT);
   glBegin(GL_TRIANGLES);
   draw_triangle(a,b,c,6);
   glEnd();
   glFlush();
}
int main(int argc,char **argv)
{
   glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_RGB|GLUT_SINGLE);
   glutInitWindowPosition(0,0);
   glutCreateWindow("Spski Gasket");
   glutDisplayFunc(display);
   myInit();
   glutMainLoop();
   return 0;
}
```

Output:

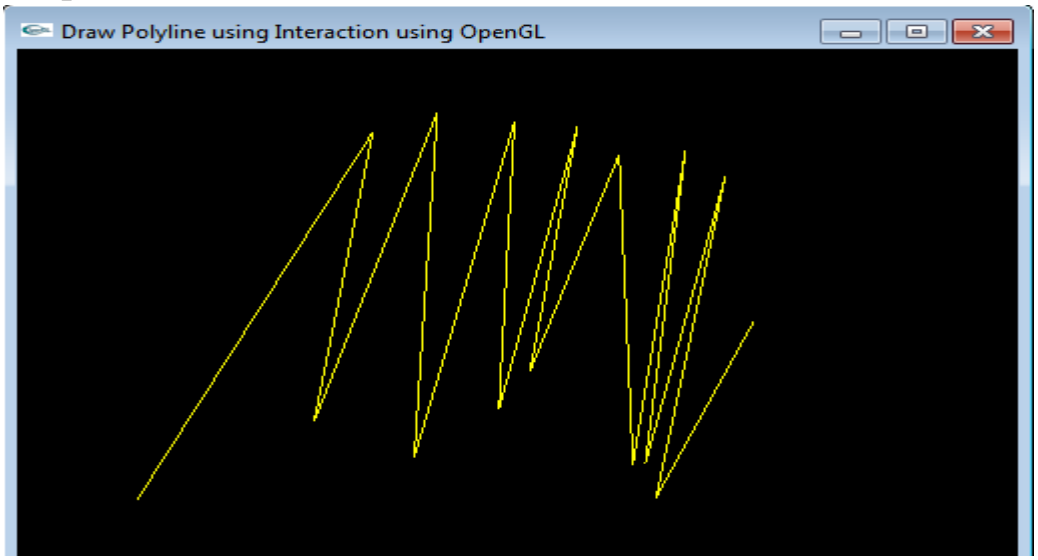


2. Create a polyline using mouse interaction using openGL*.

```
#include<GL/gl.h>
#include<GL/glut.h>
struct GLintPoint
{
 GLint x,y;
};
int Height=650,Width=650;
void myMouse(int button,int state,int x,int y);
void display(void)
{
  glClear(GL_COLOR_BUFFER_BIT);
  glFlush();                                    //Send all output to display
}
void myinit()
{
glClearColor(0.0,0.0,0.0,1.0);                  //Set background as black
glColor3f(1.0,1.0,0.0);                         //Draw in Yellow
glMatrixMode(GL_PROJECTION);                    //Establish the coordinate system
glLoadIdentity();
gluOrtho2D(0.0,650.0,0.0,650.0);
}
 void myKeyboard(unsigned char key,int mouseX,int mouseY)
{
  switch(key)
{
  case 27:
 exit(0);
 }
}
 void myMouse(int button,int state,int x,int y)
{
 static GLintPoint vertex[1];
static int pt=0;
if(button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
{
 if(pt == 0)
{
 vertex[pt].x = x;
 vertex[pt].y = Height-y;
 pt++;
}
else
{
```

```
 glBegin(GL_LINE_STRIP);
 glVertex2i(vertex[0].x,vertex[0].y);
 glVertex2i(x,Height-y);
 glEnd();
 vertex[0].x = x;
 vertex[0].y = Height-y;
}
}
glFlush();
}
int main(int argc,char**argv)
{
 glutInit(&argc,argv);                          //Initialize the toolkit
 glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);     //Set display mode
glutInitWindowSize(500,500);            // Set window size
glutInitWindowPosition(100,100);                //Set window position on the screen
// Open the screen window
glutCreateWindow("Draw Polyline using Interaction using OpenGL");
glutDisplayFunc(display);                //Register redraw function
glutKeyboardFunc(myKeyboard);      //Register keyboard function
glutMouseFunc(myMouse);              //Rigister mouse function
myinit();
glutMainLoop();                         //Go into a perpectual loop
return 0;
}
```

Output:

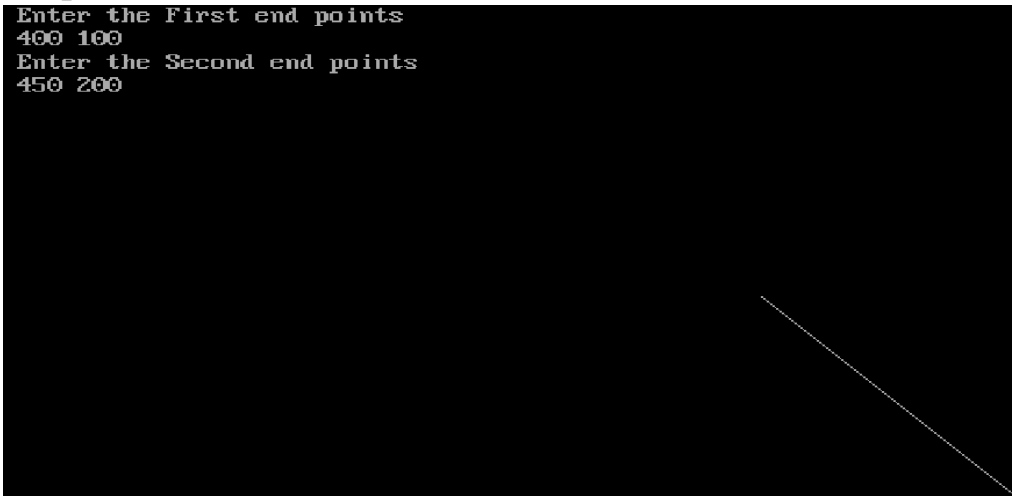


3. Bresenham's line drawing algorithm.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
void main()
{
 int xa,ya,xb,yb;
 int gd = DETECT,gmode;
 void bresenhm(int,int,int,int);
 initgraph(&gd,&gmode,"c:\\turboc3\\BGI");
 printf("Enter the First end points\n");
 scanf("%d %d",&xa,&ya);
 printf("Enter the Second end points\n");
 scanf("%d %d",&xb,&yb);
 bresenhm(xa,ya,xb,yb);
}
 void bresenhm(int xa,int ya,int xb, int yb)
 {
  int dx,dy,x,y,xend,p;
  dx= abs(xa-xb);
  dy= abs(ya-yb);
  p= 2*dy-dx;
  if(xa<xb)
  {
   x=xa;
   y=yb;
  }
  else
  {
   x=xb;
   y=yb;
   xend=xa;
  }
  putpixel(x,y,7);
  while(x<=xend)
  {
   ++x;
   if(p<0)
   {
    p+=2*dy;
   }
   else
   {
```

```
    ++y;
    p+=2*dy-dx;
   }
   putpixel(x,y,7);
  }
  getch();
}
```

Output:

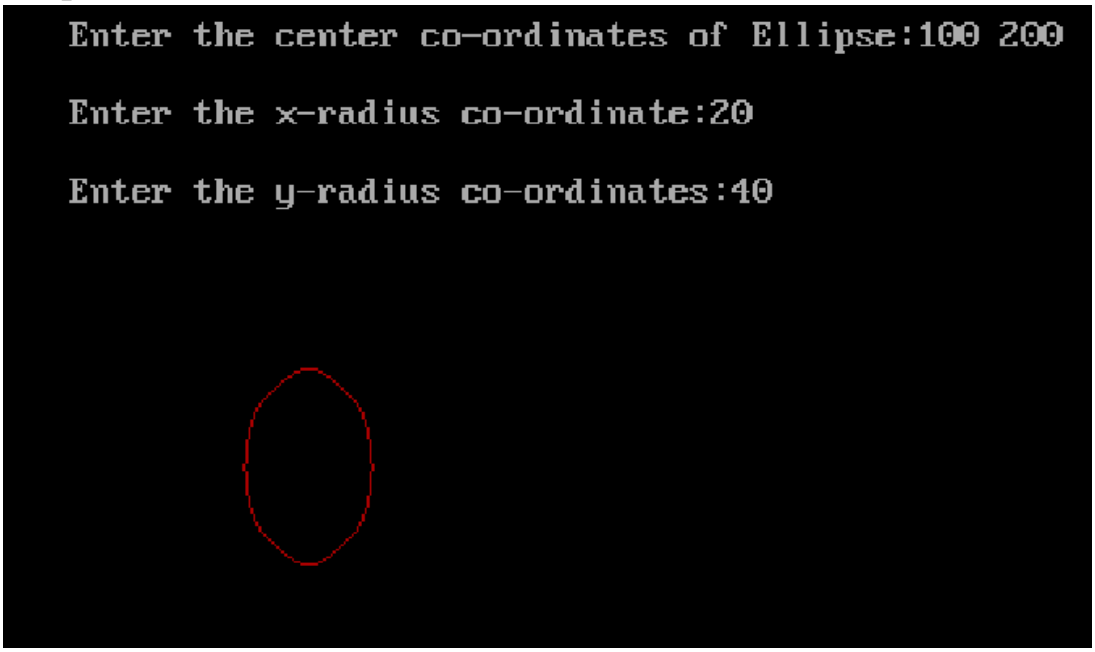

Enter the First end points
400 100
Enter the Second end points
450 200

4. Mid-Point ellipse drawing algorithm.

```
#include<stdio.h>
#include<conio.h>
#include<dos.h>
#include<graphics.h>
void ellipsemidpoint(float,float,float,float);
void drawellipse(float,float,float,float);
void main()
{
 float xc,yc,rx,ry;
 int gd=DETECT,gm;
 initgraph(&gd,&gm,"c:\\turboc3\\BGI");
 printf("\n Enter the center co ordinates of ellipse:");
 scanf("%f %f",xc,yc);
 printf("\n Enter the x radius co ordinates:");
 scanf("%f",&rx);
 printf("\n Enter the y radius co ordinates:");
 scanf("%f",&ry);
 ellipsemidpoint(xc,yc,rx,ry);
 getch();
 }
 void ellipsemidpoint(float xc,float yc,float rx,float ry)
 {
  float rxsq=rx*rx;
  float rysq=ry*ry;
  float x=0,y=ry,p;
  float px=0,py=2*rxsq*y;
  drawellipse(xc,yc,x,y);
  p=rysq-(rxsq*ry)+(0.25*rxsq);
  while(px<py)
  {
   x++;
   px=px+2*rysq;
   if(p<0)
   p=p+rysq+px;
   else
   {
    y--;
    py=py-2*rxsq;
    p=p+rysq+px-py;
   }
   drawellipse(xc,yc,x,y);
   delay(30);
  }
```

```
 //Region 2
 p=rysq*(x+0.5)+(x+0.5)+rxsq*(y-1)*(y-1)-rxsq*rysq;
 while(y>0)
 {
  y--;
  py=py-2*rxsq;
  if(p>0)
  p=(p+rxsq-py);
  else
  {
  x++;
  px=px+2+rysq;
  p=p+rxsq-py+px;
 }
 drawellipse(xc,yc,x,y);
 delay(30);
 }
}
void drawellipse(float xc,float yc,float  x,float 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);
}
```

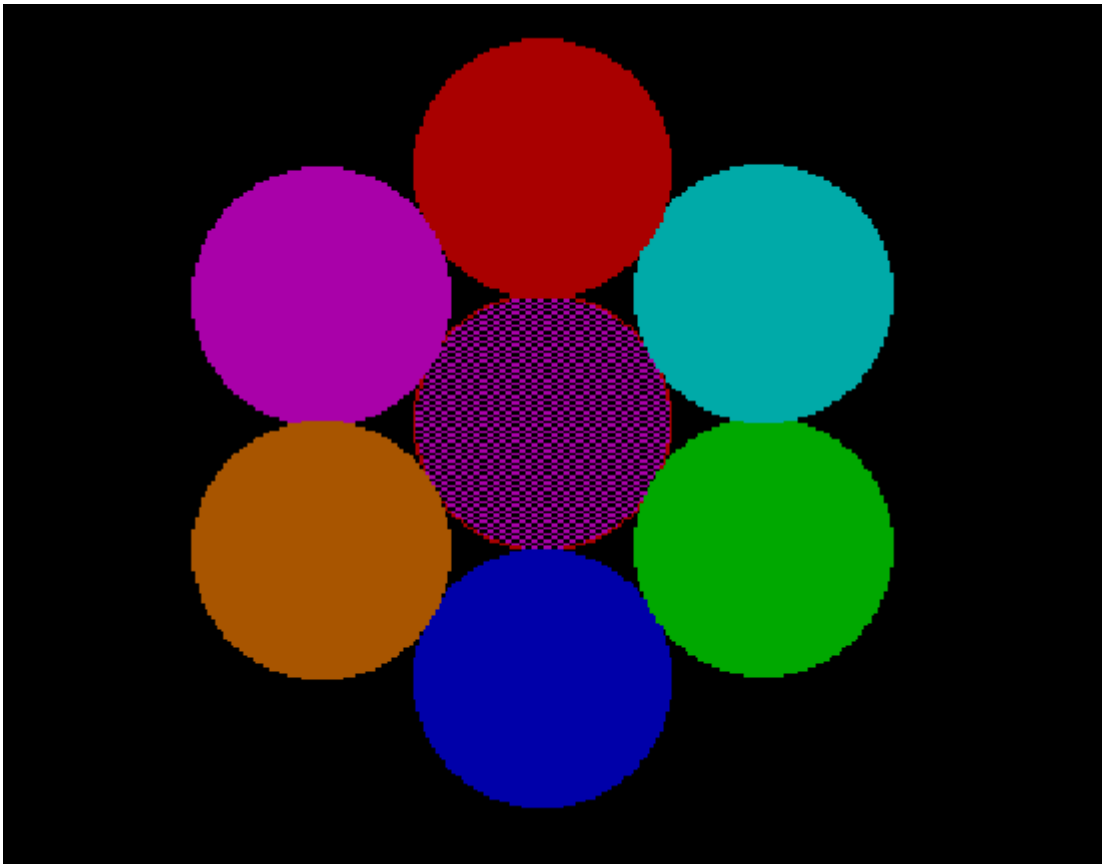
Output:


Enter the center co-ordinates of Ellipse:100 200
Enter the x-radius co-ordinate:20
Enter the y-radius co-ordinates:40

5. Implementation of Area Filling Algorithm: Boundary Fill , Flood Fill and Scan line Polygon Fill.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
 int gd=DETECT,gmode,i,xc,yc,tx,ty,nxc,nyc;
 initgraph(&gd,&gmode,"C:\\turboc3\\BGI");
 xc=getmaxx()/2;
 yc=getmaxy()/2;
 setcolor(4);
 circle(xc,yc,40);
 setfillstyle(INTERLEAVE_FILL,5);
 floodfill(xc,yc,4);
 delay(500);
 for(i=0;i<6;i++)
 {
 tx=80*sin(i*60*3.142/180);
 ty=80*cos(i*60*3.142/180);
 nxc=xc+tx;
 nyc=yc+ty;
 setcolor(i+1);
 circle(nxc,nyc,40);
 setfillstyle(SOLID_FILL,i+1);
 floodfill(nxc,nyc,i+1);
 delay(500);
 }
  getch();
}
```

Output:



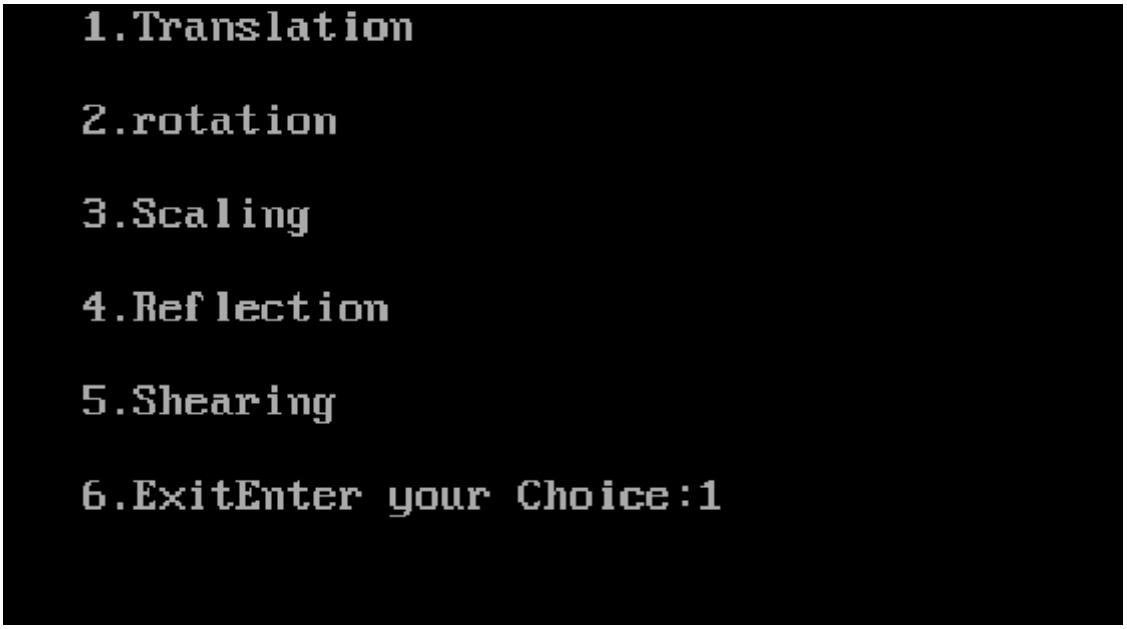
6.Program for performing Two Dimensional Transformations Translation, Scaling, Rotation, Reflection, Shear by using a homogeneous Matrix representation,use of a function for matrix multiplication is desirable, so as to perform composite transformation

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
#include<string.h>
void main()
{
 int gd=DETECT,gm,ch;
 initgraph(&gd,&gm,"c:\\turboc3\\BGI");
 cleardevice();
 printf("\t 1.Translation\n\n\t 2.rotation\n\n\t 3.Scaling\n\n\t 4.Reflection\n\n\t
5.Shearing\n\n\t 6.Exit");
 printf("Enter your Choice:");
 scanf("%d",&ch);
 switch(ch)
 {
  case 1:{
            int tx=50,ty=50,x1=100,x2=230,y1=100,y2=70;
            cleardevice();
            printf("Rectangle before Translation:\n");
            rectangle(x1,y1,x2,y2);
            getch();
            cleardevice();
            printf("Rectangle after Translation:\n");
            rectangle(x1+tx,y1+ty,x2+tx,y2+ty);
            getch();
            main();
           }
  case 2:{
          long x1=100,y1=100,x2=200,y2=200;
          double d1,xt,yt;
          cleardevice();
          printf("n\n\n\t Enter angle of Rotation:");
          scanf("%lf",&d1);
          d1=((d1*3.142)/180.0);
          xt=x1+((x2-x1)*cos(d1)-(y2-y1)*sin(d1));
          yt=y1+((x2-x1)*sin(d1)+(y2-y1)*cos(d1));
          line(x1,y1,x2,y2);
          getch();
          main();
```

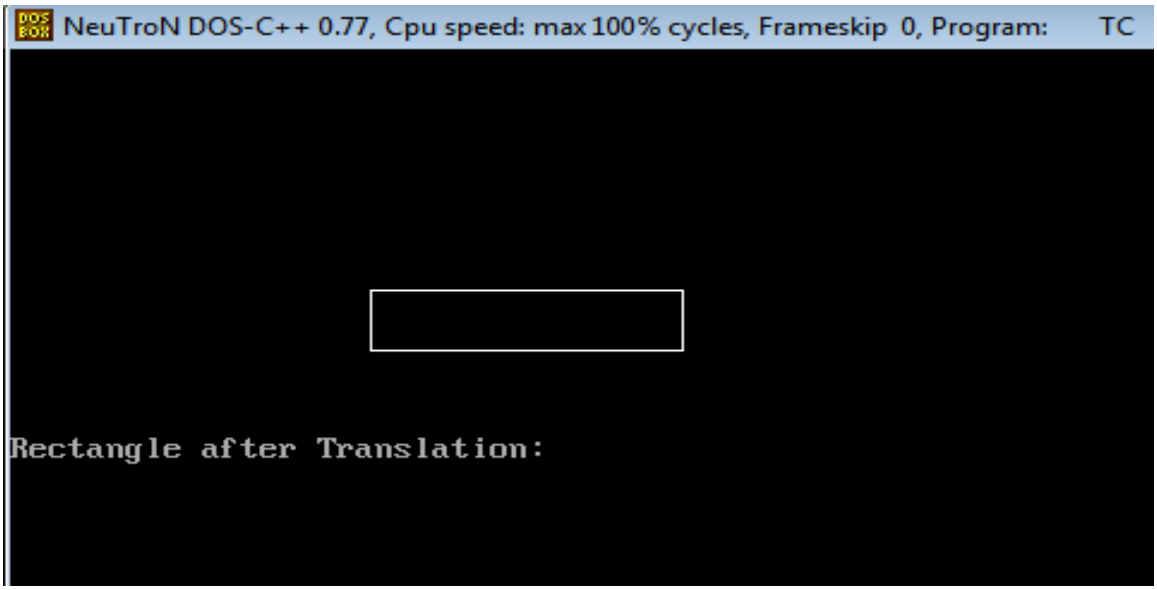
```
        }
case 3:{
        int x1=30,y1=30,x2=70,y2=70,y=2,x=2;
        cleardevice();
        printf("\n Rectangle before Scaling:\n");
        rectangle(x1,y1,x2,y2);
        getch();
        cleardevice();
        printf("\n\n\n rectangle after Scaling:\n");
        rectangle(x1*x,y1*y,x2*x,y2*y);
        getch();
        main();
        }
case 4:{
        int x1=50,y1=150,x2=75,y2=125,x3=100,y3=150,xt;
        cleardevice();
        printf("\n\n\n Triangle before Reflecation");
        line(x1,y1,x2,y2);
        line(x1,y1,x2,y2);
        line(x1,y1,x2,y2);
        getch();
        cleardevice();
        printf("\n\n\n Triangle after Reflecation\n");
        line(x1,-y1+200,x2,-y2+200);
        line(x1,-y1+200,x3,-y3+200);
        line(x2,-y2+200,x3,-y3+200);
        getch();
        main();
        }
case 5:{
        int x1=100,x2=100,y1=100,y2=30,x3=170,y3=30,x4=170,y4=30,shx;
        cleardevice();
        printf("\n\n\n Rectangle before Shearing\n");
        line(x1,y1,x2,y2);
        line(x1,y1,x4,y4);
        line(x2,y2,x3,y3);
        line(x3,y3,x4,y4);
        getch();
        cleardevice();
        printf("\n\n\n Rectangle after Shearing:\n");
        line(x1,y1,x2+shx*y2,y2);
        line(x1,y1,x4,y4);
        line(x2+shx*y2,y2,x3+shx*y3,y3);
        line(x3+shx*y3,y3,x4,y4);
```

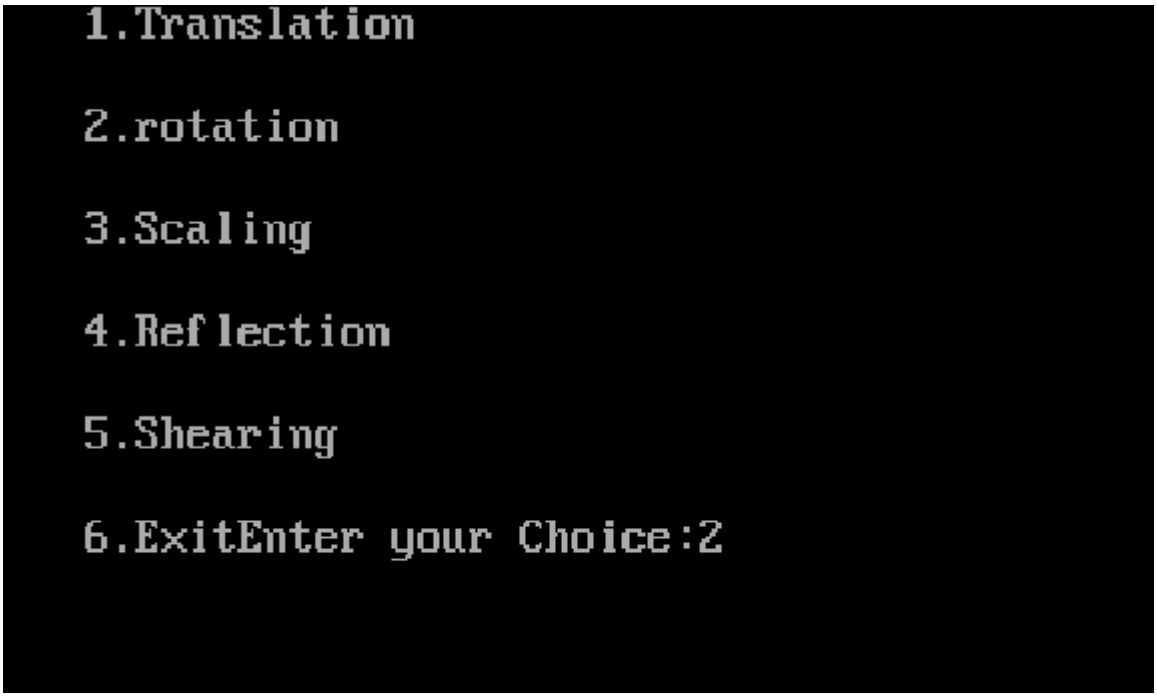
```
        getch();
        main();
        }
        }
    closegraph();
    }
```

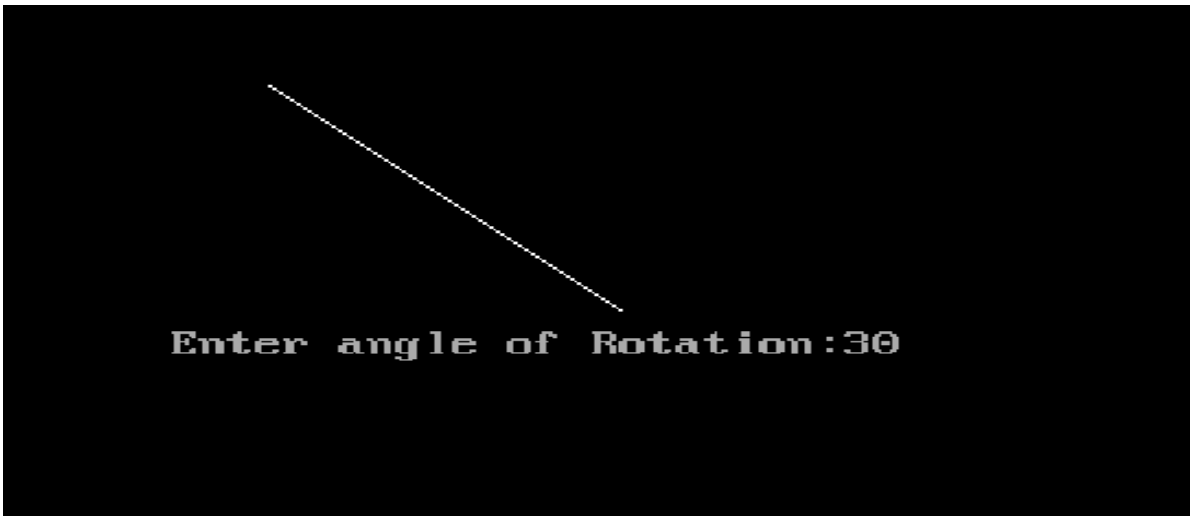
Output:


1.Translation
2.rotation
3.Scaling
4.Reflection
5.Shearing
6.ExitEnter your Choice:1

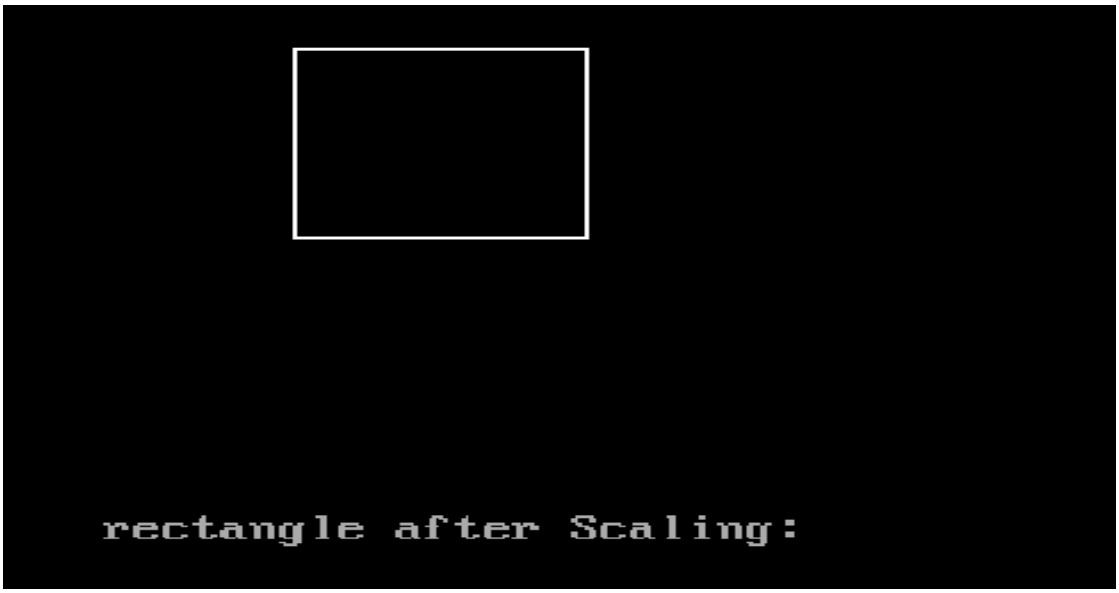

Rectangle before Translation:

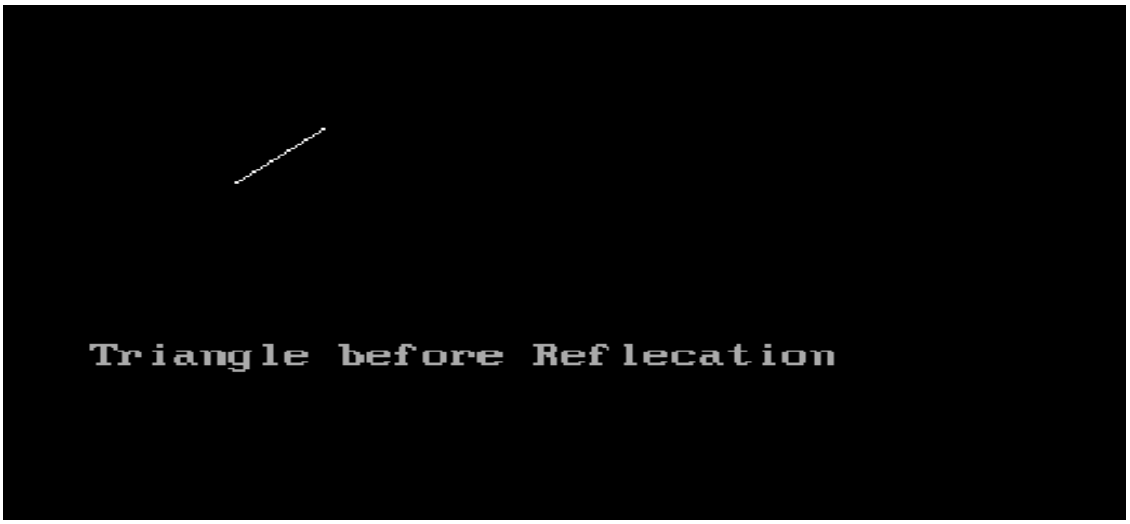

NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
Rectangle after Translation:


1.Translation
2.rotation
3.Scaling
4.Reflection
5.Shearing
6.ExitEnter your Choice:2


Enter angle of Rotation:30


Rectangle before Scaling:


rectangle after Scaling:


Triangle before Reflecation


Triangle after Reflecation

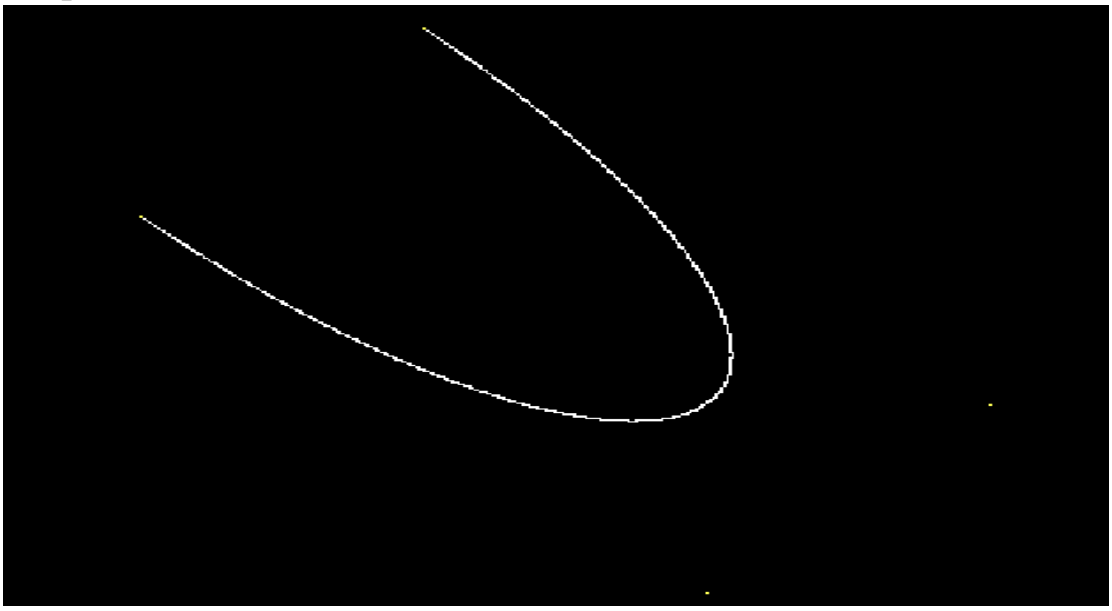

Rectangle before Shearing


Rectangle after Shearing:

7. Curve Generation : Bezier for n control points , B Spline (Uniform)

```
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
#include <math.h>
void bezier (int x[4], int y[4])
{
   int gd = DETECT, gm;
   int i;
   double t;
   initgraph (&gd,&gm,"c:\\turboc3\\BGI");
   for (t = 0.0; t < 1.0; t += 0.0005)
   {
         double xt = pow (1-t, 3) * x[0] + 3 * t * pow (1-t, 2) * x[1] +
                     3 * pow (t, 2) * (1-t) * x[2] + pow (t, 3) * x[3];

         double yt = pow (1-t, 3) * y[0] + 3 * t * pow (1-t, 2) * y[1] +
                     3 * pow (t, 2) * (1-t) * y[2] + pow (t, 3) * y[3];
         putpixel (xt, yt, WHITE);
   }
  for (i=0; i<4; i++)
         putpixel (x[i], y[i], YELLOW);
   getch();
   closegraph();
   return;
}
void main()
{
   int x[4], y[4];
   int i;
   printf ("Enter the x- and y-coordinates of the four control points.\n");
   for (i=0; i<4; i++)
         scanf ("%d%d", &x[i], &y[i]);
   bezier (x, y);
}
```

Output:



8. Line clipping algorithm Cohen-Sutherland / liang barsky.

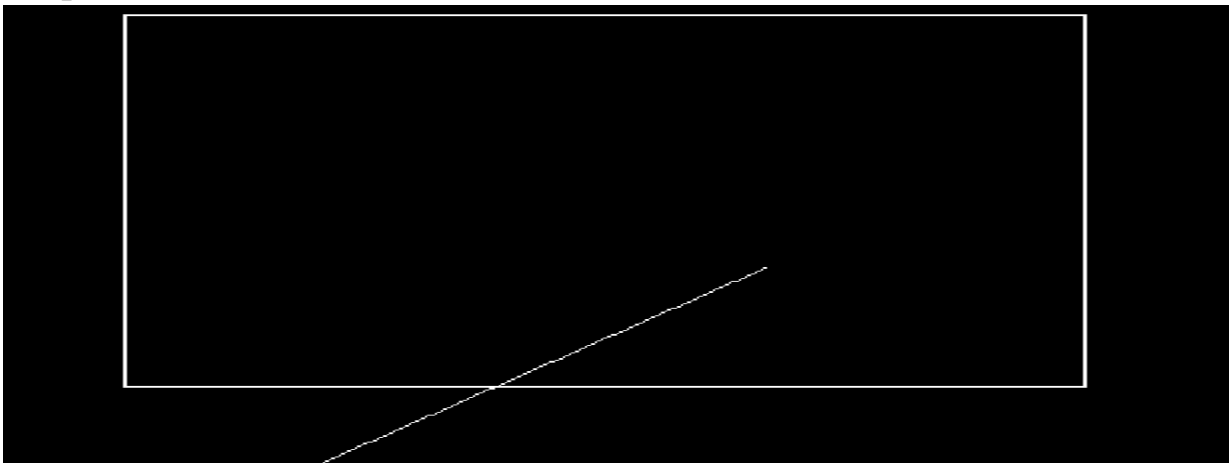
```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<dos.h>
#include<graphics.h>
typedef struct coordinate
{
 int x,y;
 char code[4];
 }
 PT;
 void drawwindow();
 void drawline(PT p1,PT p2);
 PT setcode(PT p);
 int visibility(PT p1,PT p2);
 PT resetendpt(PT p1,PT p2);

 void main()
 {
  int gd=DETECT,v,gm;
  PT p1,p2,p3,p4,ptemp;
  printf("\n Enter x1 and y1\n");
  scanf("%d%d",&p1.x,&p1.y);
  printf("\n Enter 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);
  settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
  outtextxy(150,100,"Before Clipping");
  switch(v)
  {
   case 0:
            settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
            outtextxy(150,100,"Before Clipping");
```

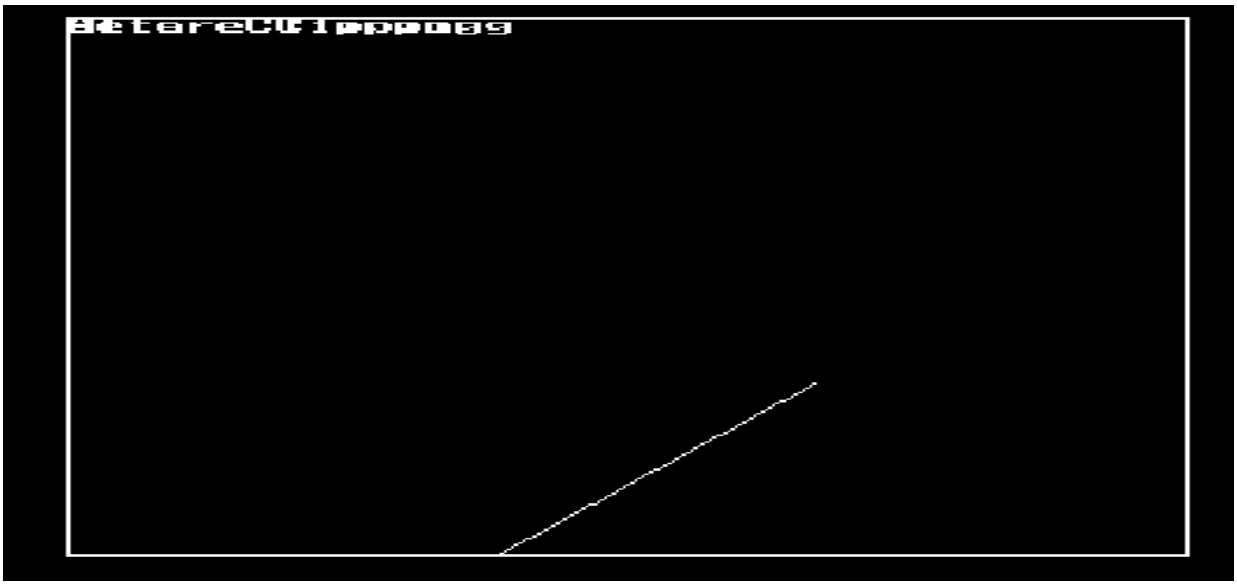
```
            drawwindow();
            drawline(p1,p2);
            delay(500);
            break;
  case 1:drawwindow();
            delay(500);
            break;
  case 2:p3=resetendpt(p1,p2);
            p4=resetendpt(p2,p1);
            drawwindow();
            delay(500);
            settextstyle(DEFAULT_FONT,HORIZ_DIR,1);
            outtextxy(150,100,"After Clipping");
            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)
  {
   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';
 else
 ptemp.code[2]='0';

 if(p.x<150)
 ptemp.code[3]='1';
 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);
    }
     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:





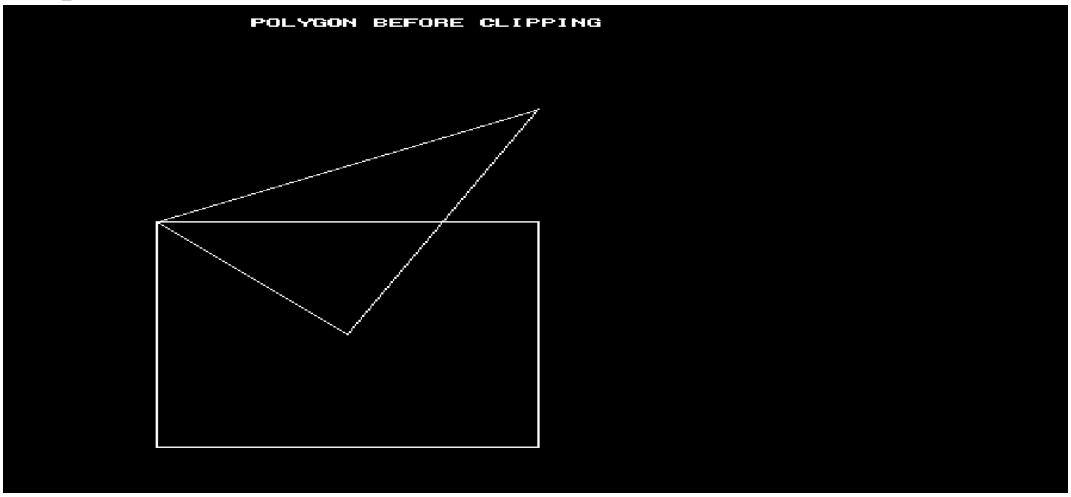
9. Polygon Clipping algorithm Sutherland Hodgeman.

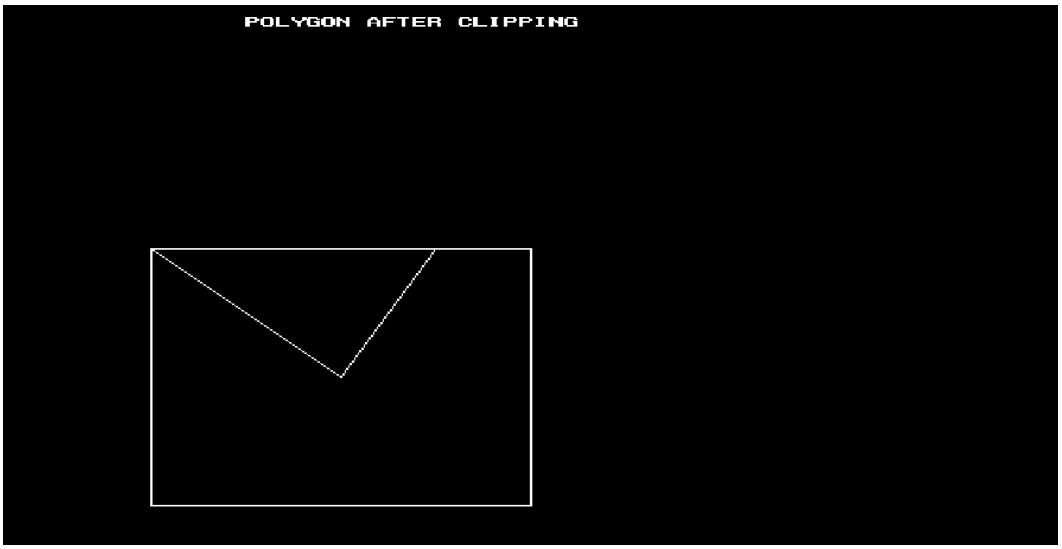
```
#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <math.h>
#include <process.h>
#define TRUE 1
#define FALSE 0
typedef unsigned int outcode;
outcode CompOutCode(float x,float y);
enum  {  TOP = 0x1,
BOTTOM = 0x2,
RIGHT = 0x4,
LEFT = 0x8
};
float xmin,xmax,ymin,ymax;
void clip(float x0,float y0,float x1,float y1)
{
outcode outcode0,outcode1,outcodeOut;
int accept = FALSE,done = FALSE;
outcode0 = CompOutCode(x0,y0);
outcode1 = CompOutCode(x1,y1);
do
{
   if(!(outcode0|outcode1))
   {
      accept = TRUE;
      done = TRUE;
   }
   else
   if(outcode0 & outcode1)
      done = TRUE;
   else
   {
      float x,y;
      outcodeOut = outcode0?outcode0:outcode1;
      if(outcodeOut & TOP)
      {
         x = x0+(x1-x0)*(ymax-y0)/(y1-y0);
         y = ymax;
      }
      else if(outcodeOut & BOTTOM)
      {
         x = x0+(x1-x0)*(ymin-y0)/(y1-y0);
```

```
        y = ymin;
       }
       else if(outcodeOut & RIGHT)
       {
          y = y0+(y1-y0)*(xmax-x0)/(x1-x0);
          x = xmax;
       }
       else
       {
          y = y0+(y1-y0)*(xmin-x0)/(x1-x0);
          x = xmin;
       }
       if(outcodeOut==outcode0)
       {
          x0 = x;
          y0 = y;
          outcode0 = CompOutCode(x0,y0);
       }
       else
       {
          x1 = x;
          y1 = y;
          outcode1 = CompOutCode(x1,y1);
        }
    }
}while(done==FALSE);
if(accept)
    line(x0,y0,x1,y1);
outtextxy(150,20,"POLYGON AFTER CLIPPING");
rectangle(xmin,ymin,xmax,ymax);
}
outcode CompOutCode(float x,float y)
{
    outcode code = 0;
    if(y>ymax)
       code|=TOP;
    else if(y<ymin)
           code|=BOTTOM;
    if(x>xmax)
       code|=RIGHT;
    else if(x<xmin)
       code|=LEFT;
    return code;
}
```

```
void main( )
{
float x1,y1,x2,y2;
/* request auto detection */
int gdriver = DETECT, gmode, n,poly[14],i;
clrscr( );
printf("Enter the no of sides of polygon:");
scanf("%d",&n);
printf("\nEnter the coordinates of polygon\n");
for(i=0;i<2*n;i++)
{
   scanf("%d",&poly[i]);
}
poly[2*n]=poly[0];
poly[2*n+1]=poly[1];
printf("Enter the rectangular coordinates of clipping window\n");
scanf("%f%f%f%f",&xmin,&ymin,&xmax,&ymax);
/* initialize graphics and local variables */
initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");

outtextxy(150,20,"POLYGON BEFORE CLIPPING");
drawpoly(n+1,poly);
rectangle(xmin,ymin,xmax,ymax);
getch( );
cleardevice( );
for(i=0;i<n;i++)
clip(poly[2*i],poly[(2*i)+1],poly[(2*i)+2],poly[(2*i)+3]);
getch( );
restorecrtmode( );
}
```

Output:





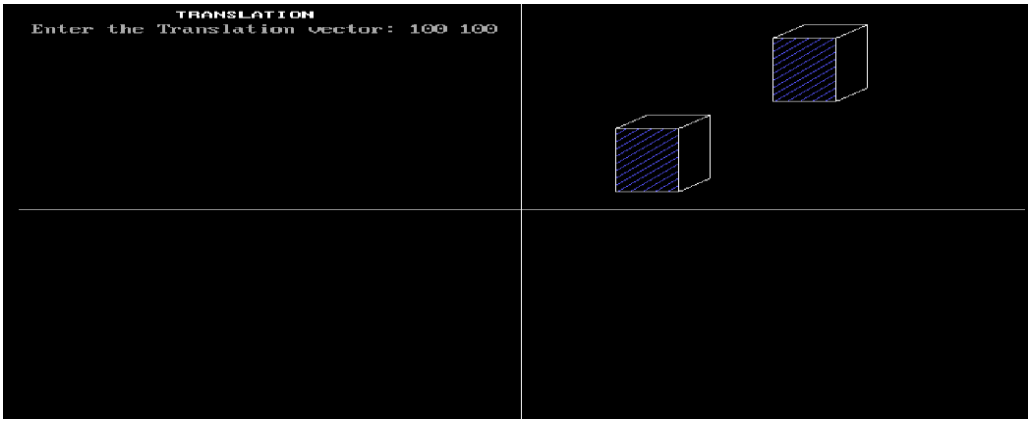
10. Program to represent a 3D object using polygon surfaces and then perform 3D transformation

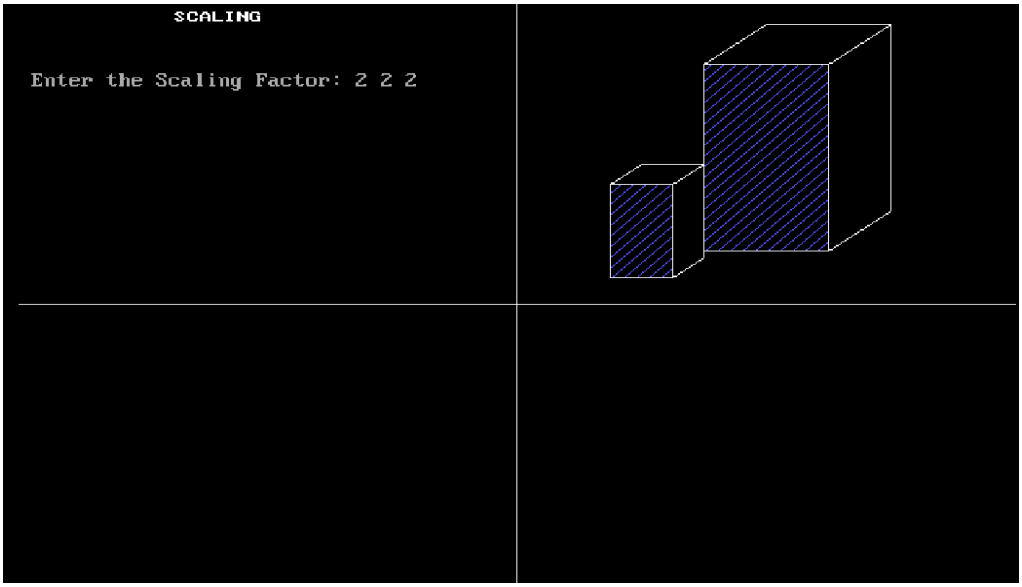
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
int maxx,maxy,midx,midy;
void axis()
{
  getch();
  cleardevice();
  line(midx,0,midx,maxy);
  line(0,midy,maxx,midy);
}
void main()
{
  int gd,gm,x,y,z,ang,x1,x2,y1,y2;
  detectgraph(&gd,&gm);
  initgraph(&gd,&gm,"C:\\turboc3\\BGI");
  setfillstyle(3,25);
  maxx=getmaxx();
  maxy=getmaxy();
  midx=maxx/2;
  midy=maxy/2;
  outtextxy(100,100,"ORIGINAL OBJECT");
  bar3d(midx+100,midy-20,midx+60,midy-90,20,5);
  axis();
  outtextxy(100,20,"TRANSLATION");
  printf("\n\n Enter the Translation vector: ");
  scanf("%d%d",&x,&y);
  bar3d(midx+100,midy-20,midx+60,midy-90,20,5);
  bar3d(midx+(x+100),midy-(y+20),midx+(x+60),midy-(y+90),20,5);
  axis();
  outtextxy(100,20,"SCALING");
  printf("\n Enter the Scaling Factor: ");
  scanf("%d%d%d",&x,&y,&z);
  bar3d(midx+100,midy-20,midx+60,midy-90,20,5);
  bar3d(midx+(x*100),midy-(y*20),midx+(x*60),midy-(y*90),20*z,5);
  axis();
  outtextxy(100,20,"ROTATION");
  printf("\n Enter the Rotation angle: ");
  scanf("%d",&ang);
  x1=100*cos(ang*3.14/180)-20*sin(ang*3.14/180);
  y1=100*sin(ang*3.14/180)+20*sin(ang*3.14/180);
```

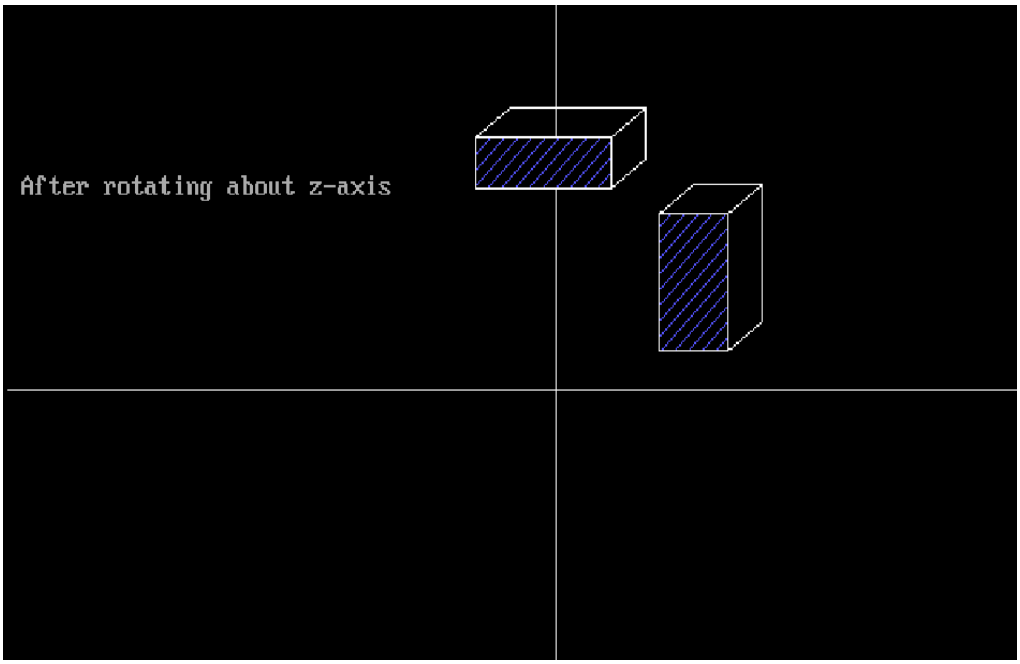
```
  x2=60*cos(ang*3.14/180)-90*sin(ang*3.14/180);
  y2=60*sin(ang*3.14/180)+90*sin(ang*3.14/180);
  axis();
  printf("\n After rotating about z-axis\n");
  bar3d(midx+100,midy-20,midx+60,midy-90,20,5);
  bar3d(midx+x1,midy-y1,midx+x2,midy-y2,20,5);
  axis();
  printf("\n After rotating about x-axis\n");
  bar3d(midx+100,midy-20,midx+60,midy-90,20,5);
  bar3d(midx+100,midy-x1,midx+60,midy-x2,20,5);
  axis();
  printf("\n After rotating about y-axis\n");
  bar3d(midx+100,midy-20,midx+60,midy-90,20,5);
  bar3d(midx+x1,midy-20,midx+x2,midy-90,20,5);
  axis();
  closegraph();
}
```

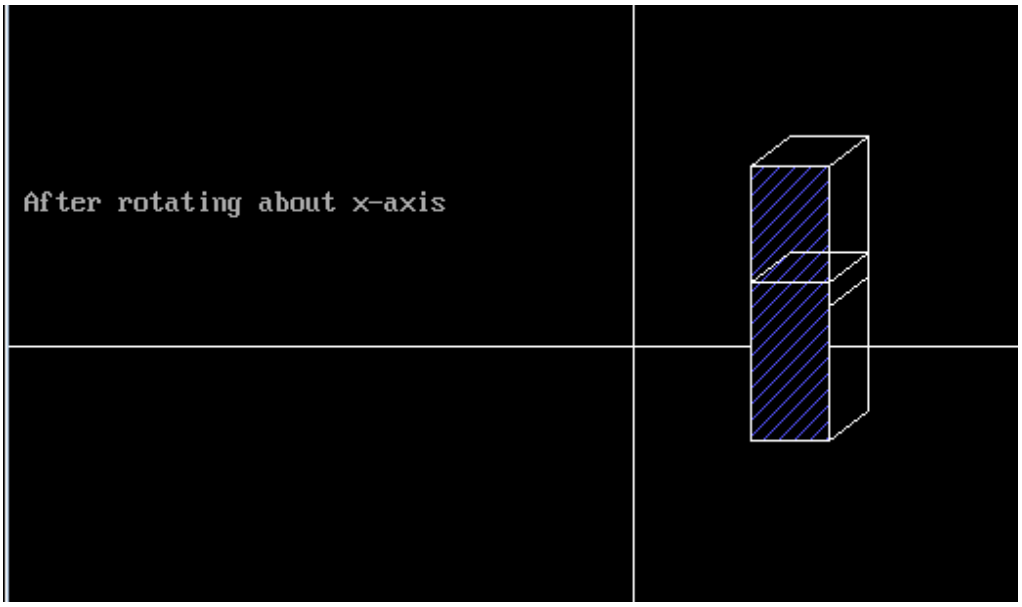
Output:

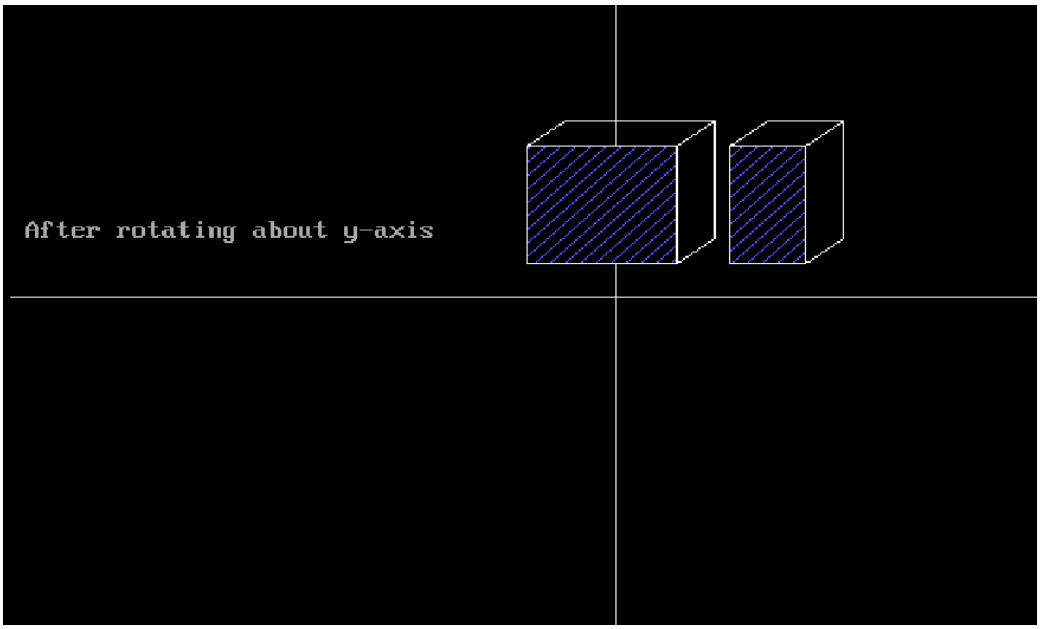

ORIGINAL OBJECT


TRANSLATION
Enter the Translation vector: 100 100


SCALING
Enter the Scaling Factor: 2 2 2


After rotating about z-axis

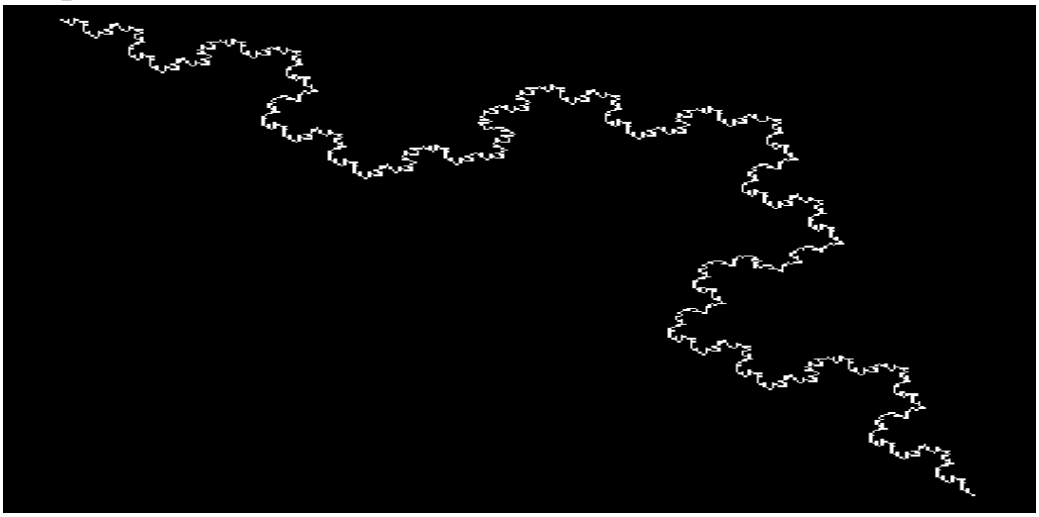

After rotating about x-axis


After rotating about y-axis

11. Fractal generation (Koch curve / Hilbert curve / peano curves using string production)

```
#include<graphics.h>
#include<conio.h>
#include<math.h>
void koch(int x1, int y1, int x2, int y2, int it)
{
 float angle = 60*M_PI/180;
 int x3 = (2*x1+x2)/3;
 int y3 = (2*y1+y2)/3;

 int x4 = (x1+2*x2)/3;
 int y4 = (y1+2*y2)/3;

 int x = x3 + (x4-x3)*cos(angle)+(y4-y3)*sin(angle);
 int y = y3 - (x4-x3)*sin(angle)+(y4-y3)*cos(angle);

 if(it > 0)
 {
  koch(x1, y1, x3, y3, it-1);
  koch(x3, y3, x, y, it-1);
  koch(x, y, x4, y4, it-1);
  koch(x4, y4, x2, y2, it-1);
 }
 else
 {
  line(x1, y1, x3, y3);
  line(x3, y3, x, y);
  line(x, y, x4, y4);
  line(x4, y4, x2, y2);
 }
}
int main(void)
{
 int gd = DETECT, gm;
 int x1=100, y1=100, x2=400, y2=400;
 initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
 koch(x1, y1, x2, y2, 4);
 getch();
 return 0;
}
```

Output:



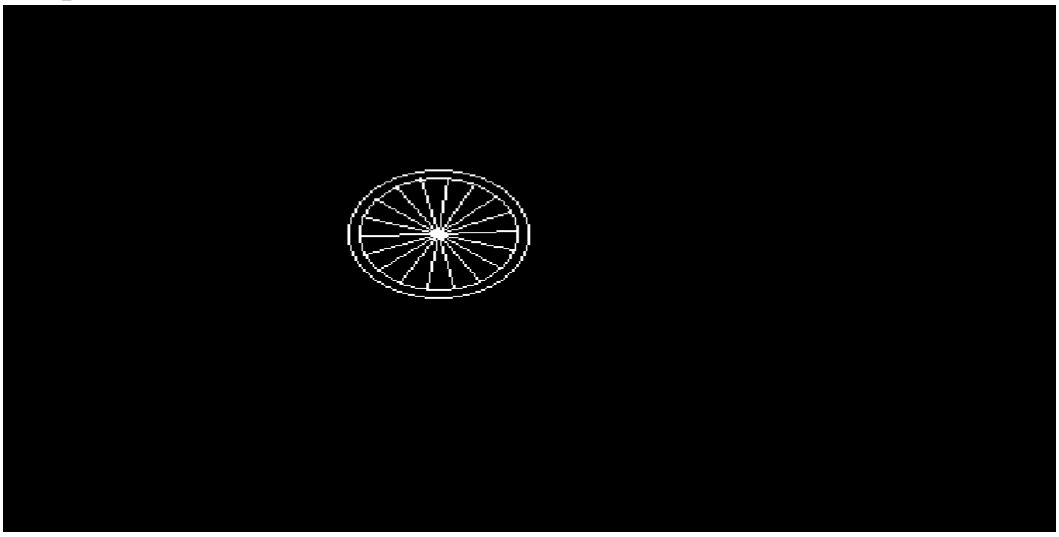
12. Program for Animation (eg. moving wheel, moving car, man walking with umbrella, flying flag, etc.)

12a. Moving Wheel

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#include<graphics.h>
#include<dos.h>
#define PI=3.142;
int xc=50,yc=200,r=35;
int x[15], y[15];
void drawCircles()
{
 setcolor(BLUE);
 circle(xc,yc,r);
 circle(xc,yc,r+5);
}
void main()
{
 double angle=0,theta;
 int i,a;
 int gd=DETECT,gm;
 initgraph(&gd,&gm,"c:\\turboc3\\BGI");
 a=xc+r;
 while(!kbhit())
 {
  while(a<=630)
  {
   theta=M_PI*angle/180;
   cleardevice();
   drawCircles();
   for(i=0;i<18;i++)
   {
    theta=M_PI*angle/180;
    x[i]=xc+r*cos(theta);
    y[i]=yc+r*sin(theta);
    angle+=20;
    line(xc,yc,x[i],y[i]);
   }
   angle+=2; xc+=2; a=xc+r;
   delay(50);
  }
  xc=50; r=35; a=xc+r;
```

```
 }
 getch();
 closegraph();
}
```

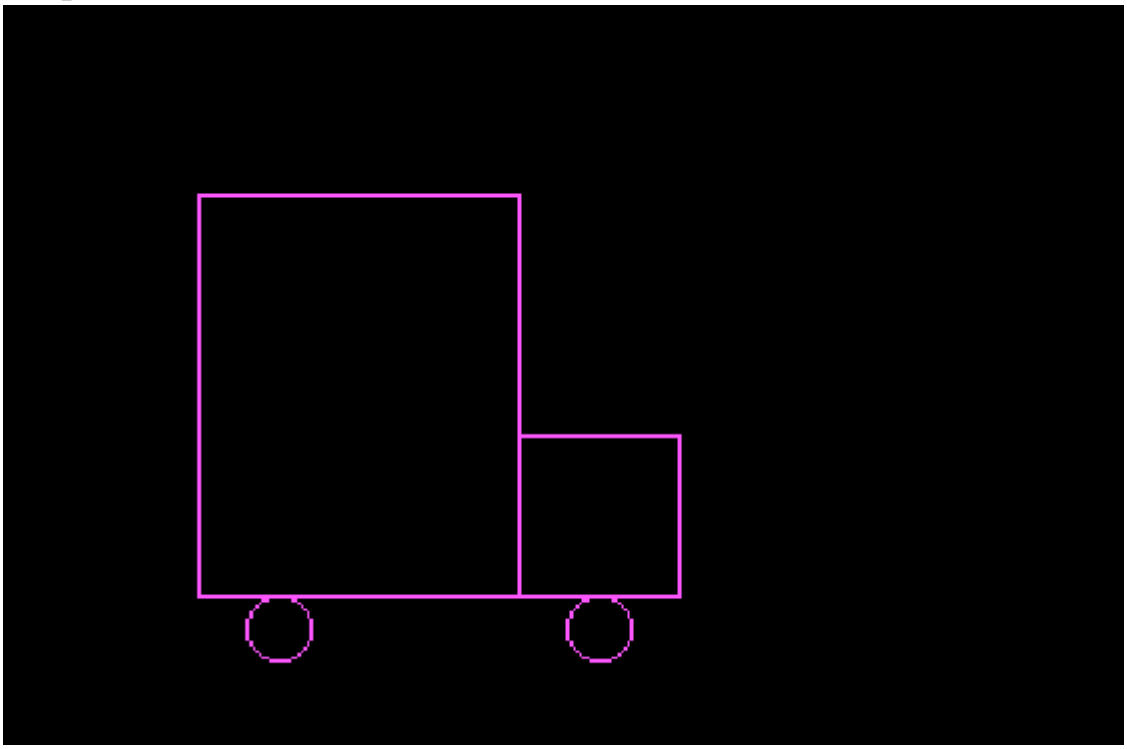
Output:



12b. MovingCar

```
#include<graphics.h>
#include<conio.h>
#include<dos.h>
main()
{
int i,j=0,gd=DETECT,gm;
initgraph(&gd,&gm,"c:\\turboc3\\BGI");
settextstyle(DEFAULT_FONT,HORIZ_DIR,2);
outtextxy(25,240,"Press any key to view the moving car");
getch();
setviewport(0,0,639,440,1);
for(i=0;i<=420;i=i+10,j++)
{
rectangle(50+i,275,150+i,400);
rectangle(150+i,350,200+i,400);
circle(75+i,410,10);
circle(175+i,410,10);
setcolor(j);
delay(100);
if(i==420)
break;
clearviewport();
}
getch();
closegraph();
return 0;
}
```

Output:



12c.Man Walking with umbrella.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void displayman(int x, int y)
{
setcolor(7);
circle(x,y,10);
line(x,y+10, x,y+30);
line(x,y+30,x-20,y+40);
line(x,y+30,x+20,y+40);
line(x+20,y+40,x+30,y+30);
line(x,y+30,x,y+70);
line(x+30,y+30,x+30,y-90);
pieslice(x+30,y-30,0,180,55);

}
void main()
{
int gd=DETECT,gm,i,d=0,j,x=50,y=340,shouldMove=1;
int rx,ry,a=1;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
while(!kbhit())
{
cleardevice();
setcolor(3);
outtextxy(100,140,"Rajanikant enjoying his first Rain ");
displayman(x,340);
line(0,430,639,430);
for(i=0; i<500; i++)
{
rx=rand()%639;
ry=rand()%439;
if(rx>=(x-40)&&rx<=(x+110))
if(ry>=(y-50)&&ry<=579)
//setcolor(2);
continue;
line(rx-10,ry+10,rx,ry);
}
if(shouldMove)
{
if(d<20)
d+=4;
else
```

```
shouldMove=0;
line(x,y+70,x-d,y+90);
line(x,y+70,x+d,y+90);
}

else
{
if(d>0)
d-=4;
else
shouldMove=1;
line(x,y+70,x-d,y+90);
line(x,y+70,x+d,y+90);

}
delay(250);
x=(x+10)%639;

}
getch();
}
```

Output:

12d. flying flag

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
 void dda(float x1,float y1,float x2,float y2,int z)
 {
        float dx,dy,x=x1,y=y1,m;
        int i;

        dx=x2-x1;
        dy=y2-y1;
        if(abs(dx)>=abs(dy))
        m=abs(dx);
        else m=abs(dy);
        putpixel((int)x,(int)y,z);
        for(i=1;i<=m;i++)
        {
                x=x+dx/m;
                y=y+dy/m;
                putpixel((int)x,(int)y,z);
        }
 }

void main()
{

        float l1=250,l2=350,l3=50,l4=80,l5=120,l6=150,i,i1,i2,i3,b,a1,a2,a3,e;

        int gd=DETECT,gm=DETECT;
        initgraph(&gd,&gm,"c:\\turboc3\\BGI");
        printf(" Algorithm DDA");
        a1=(l2-l1)/3;
        a2=(l2-l1)/2;
        a3=(l5-l4)/2;
        i1=l1+a1;
        i2=i1-l1;
        i3=i1+i2;
        setbkcolor(0);
        while(!kbhit())
        { i=1;
        e=0;

        while(((i<=i1)||(i<=i3)||(i<=l2))&&(i<=8))
        {
```

```
circle(245,47,4);
dda(240,50,240,350,8);
dda(250,50,250,350,8);
dda(350,50-i,350,150-i,7);
outtextxy(getmaxx()-150,(((getmaxy()/2)-50)+(10*i)),"JAIHIND !!!!");
outtextxy(getmaxx()-170,getmaxy()-40,"DONE BY SEENIVASAN.P");
setfillstyle(11,3);
fillellipse(l1+a2,l4+a3-i,13,13-e);
dda(l1,l3,i1,l3-i,6);
dda(l1,l4,i1,l4-i,15);
dda(l1,l5,i1,l5-i,15);
dda(l1,l6,i1,l6-i,2);

dda(i1,l3-i,i3,l3,6);
dda(i1,l4-i,i3,l4,15);
dda(i1,l5-i,i3,l5,15);
dda(i1,l6-i,i3,l6,2);

dda(i3,l3,l2,l3-i,6);
dda(i3,l4,l2,l4-i,15);
dda(i3,l5,l2,l5-i,15);
dda(i3,l6,l2,l6-i,2);

bar3d(l1-50, 355, l1+50,350+55, 10, 3);

i=i+1;
e=e+0.25;
delay(200);
cleardevice();
}
 i=8;
 b=0;
while(((i<=i1)||(i<=i3)||(i<=l1))&&(i>=1))
{
circle(245,47,4);
dda(240,50,240,350,8);
dda(250,50,250,350,8);
dda(350,50-i,350,150-i,7);
outtextxy(getmaxx()-150,(((getmaxy()/2))-50+(10*i)),"JAIHIND !!!!");
outtextxy(getmaxx()-170,getmaxy()-40,"DONE BY SEENIVASAN.P");
setfillstyle(11,4);
fillellipse(l1+a2,l4+a3-i,13,13-e);
dda(l1,l3,i1,l3-i,6);
dda(l1,l4,i1,l4-i,15);
dda(l1,l5,i1,l5-i,15);
dda(l1,l6,i1,l6-i,2);
dda(i1,l3-i,i3,l3,6);
```

```
        dda(i1,l4-i,i3,l4,15);
        dda(i1,l5-i,i3,l5,15);
        dda(i1,l6-i,i3,l6,2);
        dda(i3,l3,l2,l3-i,6);
        dda(i3,l4,l2,l4-i,15);
        dda(i3,l5,l2,l5-i,15);
        dda(i3,l6,l2,l6-i,2);
         bar3d(l1-50, 355, l1+50,350+55, 10, 3);
        i=i-1;
        e=e-0.25;
        delay(200);
        cleardevice();
        }
        }
        getch();

}
```

Output :