Index

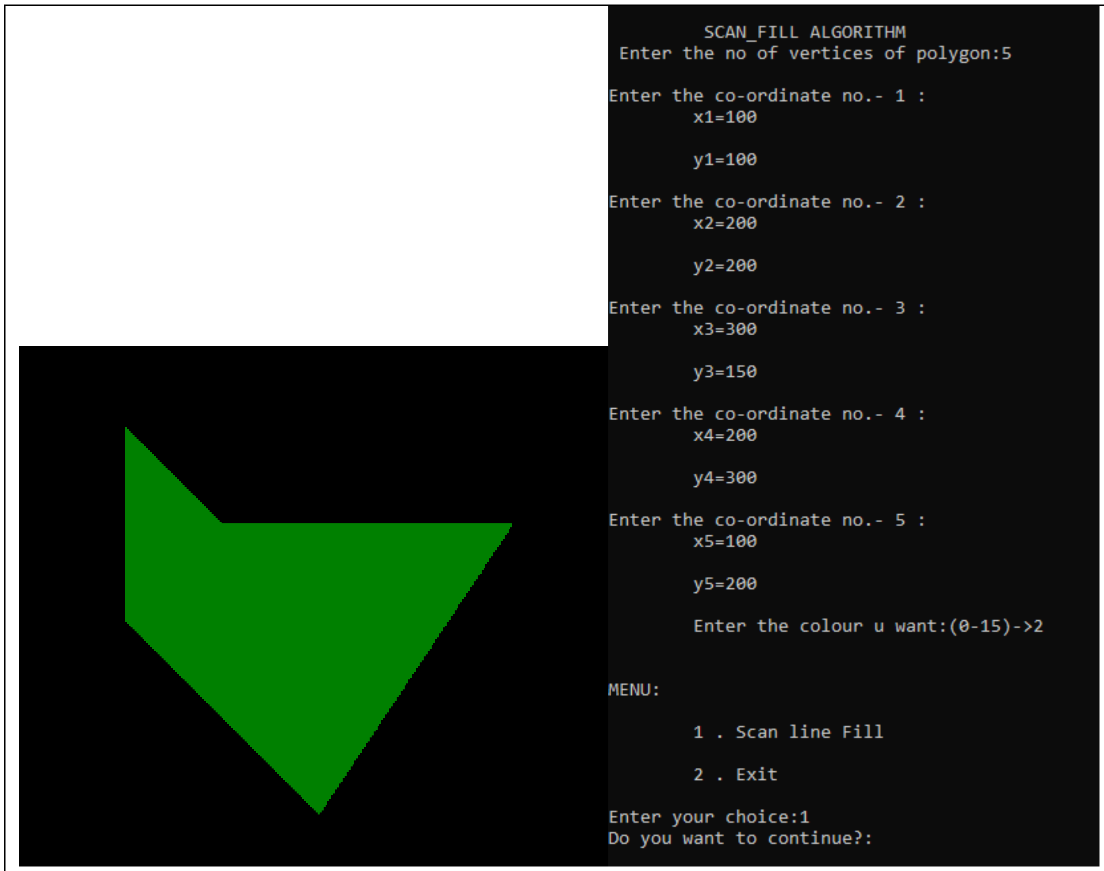
Experiment-9

Aim : Write a program to draw a concave polygon and fill it with desired color using scan fill algorithm.

```
#include <conio.h>
#include <iostream>
#include <graphics.h>
#include <stdlib.h>
using namespace std;
class point
{
   public:
   int x,y;
};
class poly{
   private:
      point p[20];
      int inter[20],x,y;
      int v,xmin,ymin,xmax,ymax;
   public:
      int c;
      void read();
      void calcs();
      void display();
      void ints(float);
      void sort(int);
};
void poly::read(){
   int i;
   cout<<"\n\t SCAN_FILL ALGORITHM";
   cout<<"\n Enter the no of vertices of polygon:";
   cin>>v;
   if(v>2)
   {
      for(i=0;i<v; i++)
      {
         cout<<"\nEnter the co-ordinate no.- "<<i+1<<" : ";
         cout<<"\n\tx"<<(i+1)<<"=";
         cin>>p[i].x;
         cout<<"\n\ty"<<(i+1)<<"=";
         cin>>p[i].y;
      }
      p[i].x=p[0].x;
      p[i].y=p[0].y;
      xmin=xmax=p[0].x;
      ymin=ymax=p[0].y;
   }
   else
      cout<<"\n Enter valid no. of vertices.";
}
void poly::calcs()
{ //MAX,MIN
   for(int i=0;i<v;i++)
   {
      if(xmin>p[i].x)
      xmin=p[i].x;
      if(xmax<p[i].x)
      xmax=p[i].x;
      if(ymin>p[i].y)
      ymin=p[i].y;
      if(ymax<p[i].y)
      ymax=p[i].y;
   }
}
void poly::display()
{
   int ch1;
   char ch='y';
   float s,s2;
   do
   {
      cout<<"\n\nMENU:";
      cout<<"\n\n\t1 . Scan line Fill ";
      cout<<"\n\n\t2 . Exit ";
      cout<<"\n\nEnter your choice:";
      cin>>ch1;
      switch(ch1){
         case 1:
            s=ymin+0.01;
            delay(100);
            cleardevice();
            while(s<=ymax)
            {
               ints(s);
               sort(s);
               s++;
            }
            break;
```

```
          case 2:
              exit(0);  }
        cout<<"Do you want to continue?: ";
        cin>>ch;
    }while(ch=='y' || ch=='Y');
}
void poly::ints(float z) {
    int x1,x2,y1,y2,temp;
    c=0;
    for(int i=0;i<v;i++){
        x1=p[i].x;
        y1=p[i].y;
        x2=p[i+1].x;
        y2=p[i+1].y;
        if(y2<y1) {
            temp=x1;
            x1=x2;
            x2=temp;
            temp=y1;
            y1=y2;
            y2=temp; }
        if(z<=y2&&z>=y1){
            if((y1-y2)==0)
            x=x1;
            else
            {
                x=((x2-x1)*(z-y1))/(y2-y1);
                x=x+x1;
            }
            if(x<=xmax && x>=xmin)
            inter[c++]=x;
        }}}
void poly::sort(int z) {
    int temp,j,i;

        for(i=0;i<v;i++)
        {
            line(p[i].x,p[i].y,p[i+1].x,p[i+1].y);
        }
        delay(100);
        for(i=0; i<c;i+=2)
        {
            delay(100);
            line(inter[i],z,inter[i+1],z);
      }}
int main() {
    int cl;
    initwindow(500,600);
    cleardevice();
    poly x;
    x.read();
    x.calcs();
    cleardevice();
    cout<<"\n\tEnter the colour u want:(0-15)->"; //Selecting colour
    cin>>cl;
    setcolor(cl);
    x.display();
    closegraph();
    getch();
    return 0;
}
```

output



```
         SCAN_FILL ALGORITHM
 Enter the no of vertices of polygon:5

Enter the co-ordinate no.- 1 :
        x1=100

        y1=100

Enter the co-ordinate no.- 2 :
        x2=200

        y2=200

Enter the co-ordinate no.- 3 :
        x3=300

        y3=150

Enter the co-ordinate no.- 4 :
        x4=200

        y4=300

Enter the co-ordinate no.- 5 :
        x5=100

        y5=200

        Enter the colour u want:(0-15)->2


MENU:

        1 . Scan line Fill

        2 . Exit

Enter your choice:1
Do you want to continue?:
```

Experiment 10

Aim : Write a program to implement Cohen Southerland line clipping algorithm.

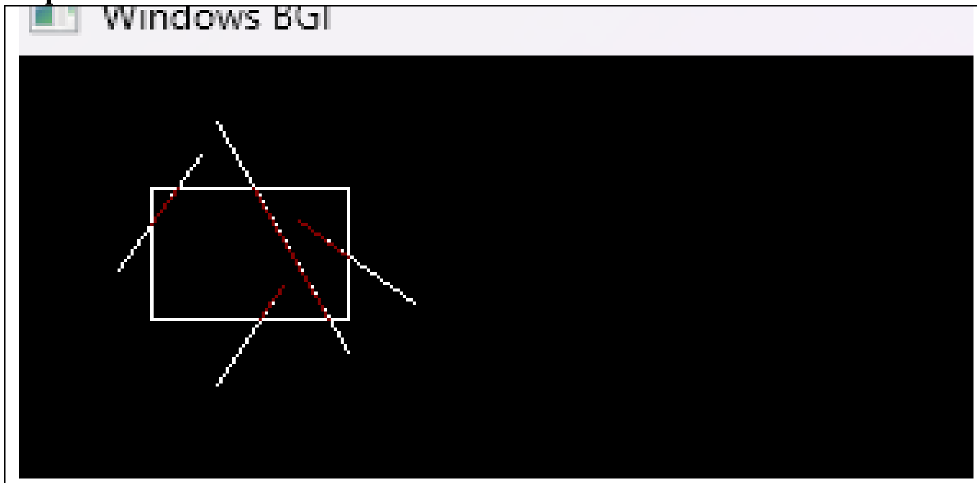
input

```
#include <bits/stdc++.h>
#include <graphics.h>
using namespace std;
int xmin, xmax, ymin, ymax;
class lines {
 public:
  int x1, y1, x2, y2;
  lines() { x1 = y1 = x2 = y2 = 0; }
  void set(int a, int b, int c, int d) {
    x1 = a;
    y1 = b;
    x2 = c;
    y2 = d;
  }
};
int sign(int x) {
  if (x > 0)
    return 1;
  else
    return 0;
}
void clip(lines mylines) {
  int bits[4], bite[4], i, var;
  setcolor(RED);
  bits[0] = sign(xmin - mylines.x1);
  bite[0] = sign(xmin - mylines.x2);
  bits[1] = sign(mylines.x1 - xmax);
  bite[1] = sign(mylines.x2 - xmax);
  bits[2] = sign(ymin - mylines.y1);
  bite[2] = sign(ymin - mylines.y2);
  bits[3] = sign(mylines.y1 - ymax);
  bite[3] = sign(mylines.y2 - ymax);

  string initial = "", end = "", temp = "";

  for (i = 0; i < 4; i++) {
    if (bits[i] == 0)
      initial += '0';
```

```
   else
     initial += '1';
 }
 for (i = 0; i < 4; i++) {
   if (bite[i] == 0)
     end += '0';
   else
     end += '1';
 }

 float m = (mylines.y2 - mylines.y1) / (float)(mylines.x2 - mylines.x1);
 float c = mylines.y1 - m * mylines.x1;

 if (initial == end && end == "0000") {
   line(mylines.x1, mylines.y1, mylines.x2, mylines.y2);
   return;
 }

 else {
   for (i = 0; i < 4; i++) {
     int val = (bits[i] & bite[i]);
     if (val == 0)
       temp += '0';
     else
       temp += '1';
   }

   if (temp != "0000") return;

   for (i = 0; i < 4; i++) {
     if (bits[i] == bite[i]) continue;

     if (i == 0 && bits[i] == 1) {
       var = round(m * xmin + c);
       mylines.y1 = var;
       mylines.x1 = xmin;
     }

     if (i == 0 && bite[i] == 1) {
       var = round(m * xmin + c);
       mylines.y2 = var;
       mylines.x2 = xmin;
     }
```

```
    if (i == 1 && bits[i] == 1) {
      var = round(m * xmax + c);
      mylines.y1 = var;
      mylines.x1 = xmax;
    }

    if (i == 1 && bite[i] == 1) {
      var = round(m * xmax + c);
      mylines.y2 = var;
      mylines.x2 = xmax;
    }
    if (i == 2 && bits[i] == 1) {
      var = round((float)(ymin - c) / m);
      mylines.y1 = ymin;
      mylines.x1 = var;
    }
    if (i == 2 && bite[i] == 1) {
      var = round((float)(ymin - c) / m);
      mylines.y2 = ymin;
      mylines.x2 = var;
    }
    if (i == 3 && bits[i] == 1) {
      var = round((float)(ymax - c) / m);
      mylines.y1 = ymax;
      mylines.x1 = var;
    }
    if (i == 3 && bite[i] == 1) {
      var = round((float)(ymax - c) / m);
      mylines.y2 = ymax;
      mylines.x2 = var;
    }
    bits[0] = sign(xmin - mylines.x1);
    bite[0] = sign(xmin - mylines.x2);
    bits[1] = sign(mylines.x1 - xmax);
    bite[1] = sign(mylines.x2 - xmax);
    bits[2] = sign(ymin - mylines.y1);
    bite[2] = sign(ymin - mylines.y2);
    bits[3] = sign(mylines.y1 - ymax);
    bite[3] = sign(mylines.y2 - ymax);
  }
  initial = "", end = "";
  for (i = 0; i < 4; i++) {
      if (bits[i] == 0)
        initial += '0';
      else
        initial += '1';
    }
    for (i = 0; i < 4; i++) {
      if (bite[i] == 0)
        end += '0';
      else
        end += '1';
    }
    if (initial == end && end == "0000") {
      line(mylines.x1, mylines.y1, mylines.x2, mylines.y2);
      return;
    } else
      return;
  }
}

int main() {
  xmin = 40;
  xmax = 100;
  ymin = 40;
  ymax = 80;

  int gd = DETECT, gm;
  initgraph(&gd, &gm, NULL);

  rectangle(xmin, ymin, xmax, ymax);

  lines mylines[4];

  mylines[0].set(30, 65, 55, 30);
  mylines[1].set(60, 20, 100, 90);
  mylines[2].set(60, 100, 80, 70);
  mylines[3].set(85, 50, 120, 75);

  for (int i = 0; i < 4; i++) {
    line(mylines[i].x1, mylines[i].y1, mylines[i].x2, mylines[i].y2);
    delay(1000);
  }

  for (int i = 0; i < 4; i++) {
    clip(mylines[i]);
    delay(1000);
  }
  delay(4000);
  getch();
  closegraph();
  return 0;
}
```

output



Experiment 11

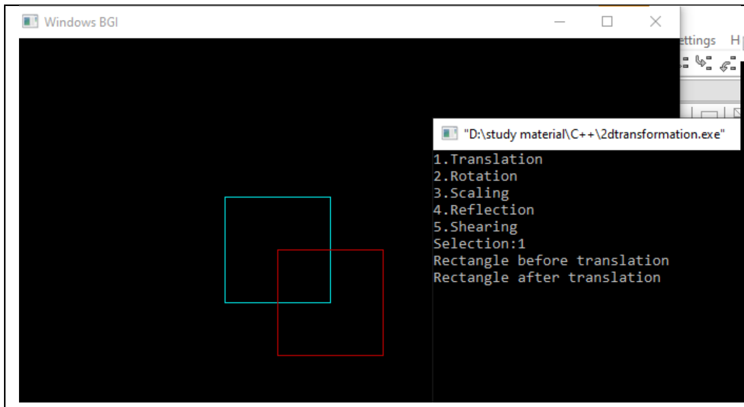
Aim : Write a program to draw 2-D object and perform following basic transformations, a) Scaling b) Translation c) Rotation. Use operator overloading.

input

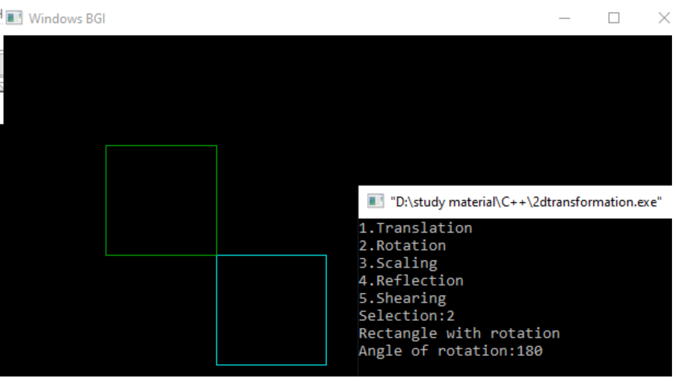
```
#include<iostream>
#include<graphics.h>
#include<math.h>
using namespace std;
int main()
{
   int gd=DETECT,gm,s;
   initgraph(&gd,&gm,(char*)"");
   cout<<"1.Translation\n2.Rotation\n3.Scaling\n4.Reflection\n5.Shearing   "<<endl;
   cout<<"Selection:";
   cin>>s;
   switch(s)
      {
      case 1:
         {   int x1=200,y1=150,x2=300,y2=250;
            int tx=50,ty=50;
            cout<<"Rectangle before translation"<<endl;
            setcolor(3);
            rectangle(x1,y1,x2,y2);
            setcolor(4);
            cout<<"Rectangle after translation"<<endl;
            rectangle(x1+tx,y1+ty,x2+tx,y2+ty);
            getch();
            break;
         }
      case 2:
         {  long x1=200,y1=200,x2=300,y2=300;
            double a;
            cout<<"Rectangle with rotation"<<endl;
            setcolor(3);
            rectangle(x1,y1,x2,y2);
            cout<<"Angle of rotation:";
            cin>>a;
            a=(a*3.14)/180;
            long xr=x1+((x2-x1)*cos(a)-(y2-y1)*sin(a));
            long yr=y1+((x2-x1)*sin(a)+(y2-y1)*cos(a));
            setcolor(2);
            rectangle(x1,y1,xr,yr);
            getch();
            break;
}
      case 3:
         {
            int x1=30,y1=30,x2=70,y2=70,y=2,x=2;
```

```
                cout<<"Before scaling"<<endl;
                setcolor(3);
                rectangle(x1,y1,x2,y2);
                cout<<"After scaling"<<endl;
                setcolor(10);
                rectangle(x1*x,y1*y,x2*x,y2*y);
                getch();
                break;}
        case 4:
            {
                int x1=200,y1=300,x2=500,y2=300,x3=350,y3=400;
                cout<<"triangle before reflection"<<endl;
                setcolor(3);
                line(x1,y1,x2,y2);
                line(x1,y1,x3,y3);
                line(x2,y2,x3,y3);
cout<<"triangle after reflection"<<endl;
                setcolor(5);
                line(x1,-y1+500,x2,-y2+500);
                line(x1,-y1+500,x3,-y3+500);
                line(x2,-y2+500,x3,-y3+500);
                getch();
                break;}
        case 5:
        {
  int x1=400,y1=100,x2=600,y2=100,x3=400,y3=200,x4=600,y4=200,shx=2;
            cout<<"Before shearing of rectangle"<<endl;
            setcolor(3);
            line(x1,y1,x2,y2);
            line(x1,y1,x3,y3);
            line(x3,y3,x4,y4);
            line(x2,y2,x4,y4);
            cout<<"After shearing of rectangle"<<endl;
            x1=x1+shx*y1;
            x2=x2+shx*y2;
            x3=x3+shx*y3;
            x4=x4+shx*y4;
            setcolor(13);
            line(x1,y1,x2,y2);
            line(x1,y1,x3,y3);
            line(x3,y3,x4,y4);
            line(x2,y2,x4,y4);
getch();}
default:
          {
            cout<<"Invalid Selection"<<endl;
            break;
          } }
closegraph();
    return 0;}
```

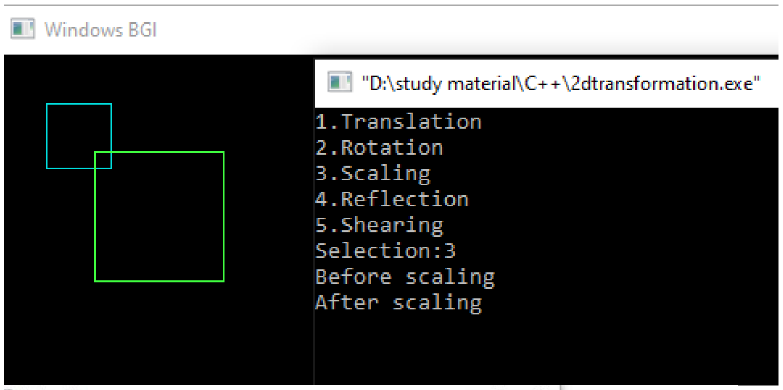
output



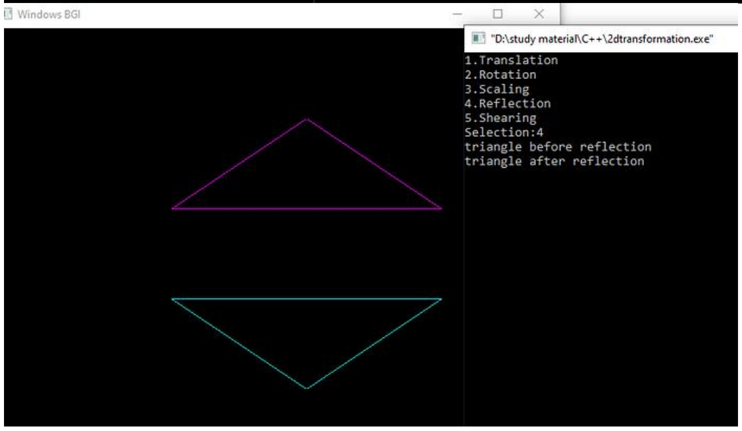
```
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing
Selection:1
Rectangle before translation
Rectangle after translation
```



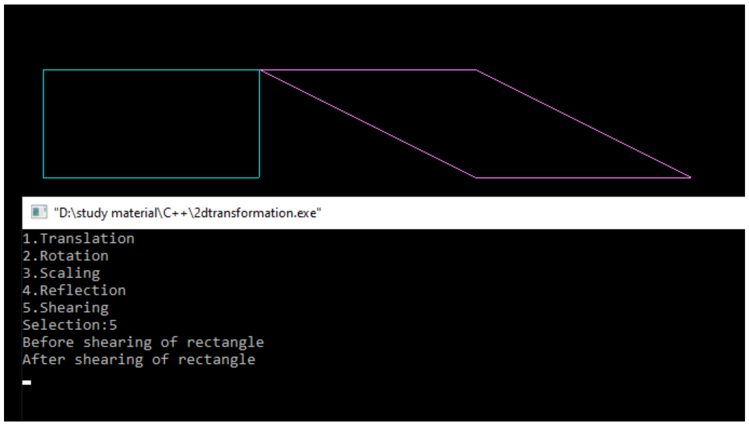
```
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing
Selection:2
Rectangle with rotation
Angle of rotation:180
```



```
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing
Selection:3
Before scaling
After scaling
```



```
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing
Selection:4
triangle before reflection
triangle after reflection
```



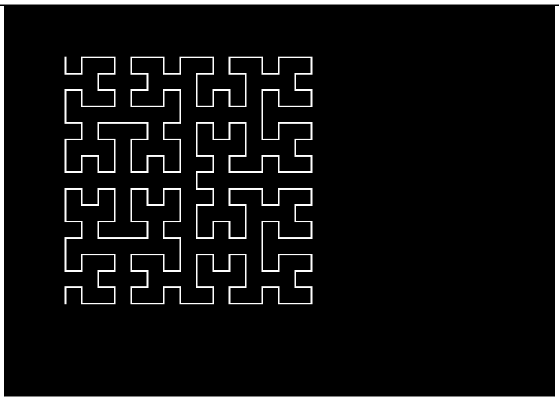
```
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing
Selection:5
Before shearing of rectangle
After shearing of rectangle
```

Experiment 12

Aim: Write a program to generate Hilbert Curve using concept of fractals.

input

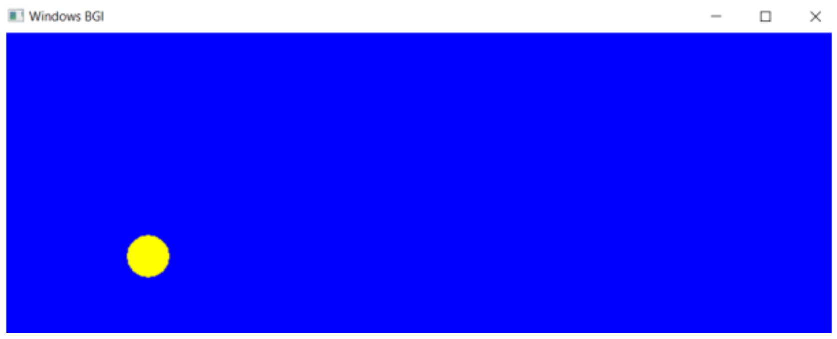
```
#include <iostream>
#include <stdlib.h>
#include <graphics.h>
#include <math.h>
using namespace std;
void move(int j,int h,int &x,int &y)
{
if(j==1)
y-=h;
else if(j==2)
x+=h;
else if(j==3)
y+=h;
else if(j==4)
x-=h;
lineto(x,y);
}
void hilbert(int r,int d,int l,int u,int i,int h,int &x,int
&y)
{
if(i>0)
{
i--;
hilbert(d,r,u,l,i,h,x,y);
move(r,h,x,y);
hilbert(r,d,l,u,i,h,x,y);
move(d,h,x,y);
hilbert(r,d,l,u,i,h,x,y);
move(l,h,x,y);
hilbert(u,l,d,r,i,h,x,y);
}}
int main(){
int n,x1,y1;
int x0=50,y0=150,x,y,h=10,r=2,d=3,l=4,u=1;
cout<<"\nGive the value of n: ";
cin>>n;
x=x0;y=y0;
int gm,gd=DETECT;
initgraph(&gd,&gm,NULL);
moveto(x,y);
hilbert(r,d,l,u,n,h,x,y);
delay(10000);
closegraph();
return 0;
}
```

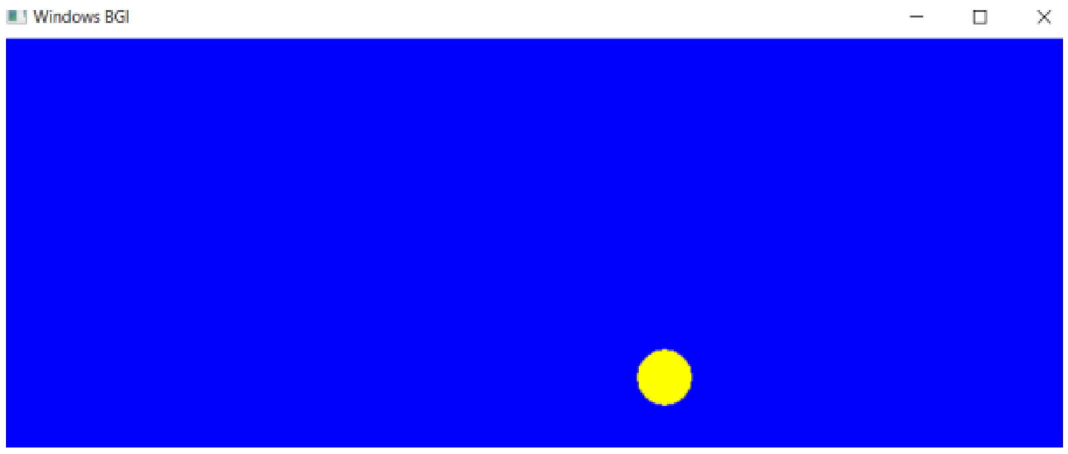


exp-13: Write a program to draw Sunrise and Sunset.

```
#include<iostream>
#include<graphics.h>
#include<cstdlib>
#include<dos.h>
#include<cmath>
using namespace std;
int main(){
 initwindow(800,500);
 int x0,y0;
 int gdriver = DETECT,gmode,errorcode;
 int xmax,ymax;
 errorcode=graphresult();
 if(errorcode!=0){
 cout<<"Graphics
error:"<<grapherrormsg(errorcode);
 cout<<"Press any ket to halt";
 exit(1);
 }
 int i,j;
 setbkcolor(BLUE);
 setcolor(RED);
 rectangle(0,0,getmaxx(),getmaxy());
 outtextxy(250,240,"::::PRESS ANY KEY TO
CONTINUE:::::");
 while(!kbhit());
 for(i=50,j=0;i<=250,j<=250;i+=5,j+=5){
 delay(120);
 cleardevice();
 if(i<=150) {
 setcolor(YELLOW);
 setfillstyle(1,YELLOW);
 fillellipse(i,300-j,20,20);
 }
 else {
 setcolor(GREEN^RED);
 setfillstyle(1,GREEN^RED);
 fillellipse(i,300-j,20,20); } }
 delay(1000);
 cleardevice();
 setcolor(RED);
 setfillstyle(1,RED);
 fillellipse(300,50,20,20);
 delay(150);
 int k,l;
 for(k=305,l=55;k<=550,l<=300;k+=5,l+=5){
 delay(120);
 cleardevice();
 if(k<=450)
 {
 setcolor(GREEN^RED);
 setfillstyle(1,GREEN^RED);
 fillellipse(k,l,20,20);
 }
 else {
 setcolor(YELLOW);
 setfillstyle(1,YELLOW);
 fillellipse(k,l,20,20); }}
 return 0;}
```

output

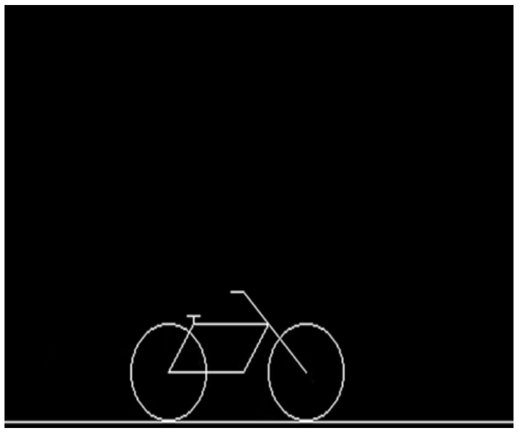




14- Draw a moving cycle using computer graphics programming in C/C++.

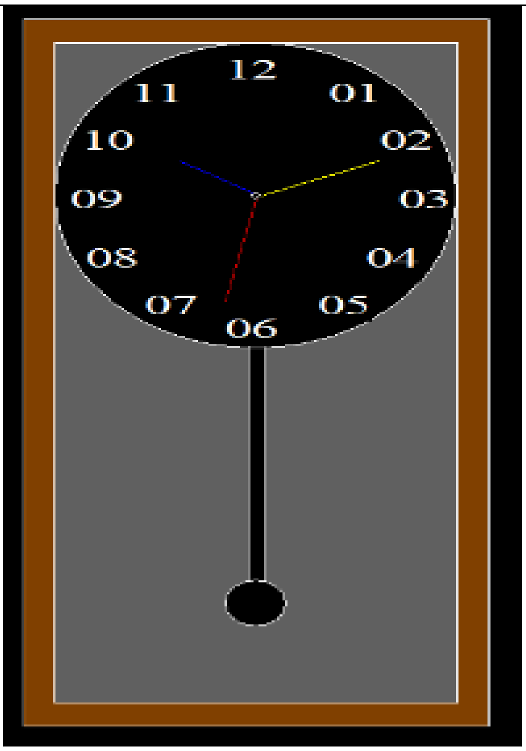
input

```
#include <conio.h>
#include <dos.h>
#include <graphics.h>
#include <iostream.h>
// Driver code
int main()
{
   int gd = DETECT, gm, i, a;
   // Path of the program
   initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
   // Move the cycle
   for (i = 0; i < 600; i++) {
      // Upper body of cycle
      line(50 + i, 405, 100 + i, 405);
      line(75 + i, 375, 125 + i, 375);
      line(50 + i, 405, 75 + i, 375);
      line(100 + i, 405, 100 + i, 345);
      line(150 + i, 405, 100 + i, 345);
      line(75 + i, 345, 75 + i, 370);
      line(70 + i, 370, 80 + i, 370);
      line(80 + i, 345, 100 + i, 345);
      // Wheel
      circle(150 + i, 405, 30);
      circle(50 + i, 405, 30);
      // Road
      line(0, 436, getmaxx(), 436);
      // Stone
      rectangle(getmaxx() - i, 436,
              650 - i, 431);
      // Stop the screen for 10 secs
      delay(10);
      // Clear the screen
      cleardevice();
   }
   getch();
   // Close the graph
   closegraph();
}
```



15- Write a program to make a digital clock using C/C++ in computer graphics.
input

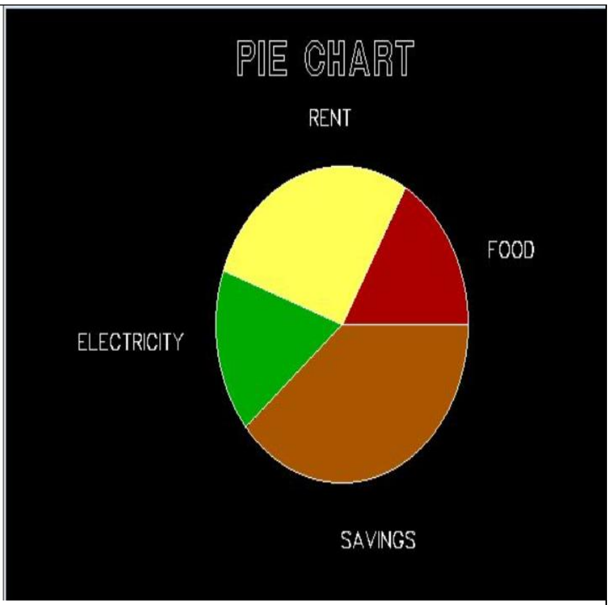
```
#include <conio.h>
#include <graphics.h>
#include <stdio.h>
// Driver Code
void main()
{
        int gd = DETECT, gm;
        // Initialize of gdriver
        initgraph(&gd, &gm, "C:\
        // Clock Outer Outline
        rectangle(500, 50, 800, 650);
        // Clock Inner Outline
        rectangle(520, 70, 780, 630);
        // Coloring Middle Part Of
        // Rectangle With Brown
        setfillstyle(SOLID_FILL, BROWN);
        floodfill(505, 55, 15);
        // Clock Outline
        circle(650, 200, 130);
        circle(650, 200, 3);
        // Coloring all the parts Of the
        // clock except the circle with
        // Darkgray

        line(647, 197, 600, 170);
        // Creating Minute Hand
        // & Color Yellow
        setcolor(YELLOW);
        line(653, 200, 730, 170);
        // Creating Second Hand and the
        // Color Red
        setcolor(RED);
        line(650, 203, 630, 290);
        // Hold the screen for a whi
        // Close the initialized gdriver
        closegraph();
}
```



exp-16 Write a program to draw a Pie Chart using C/C++ in Computer Graphics

input

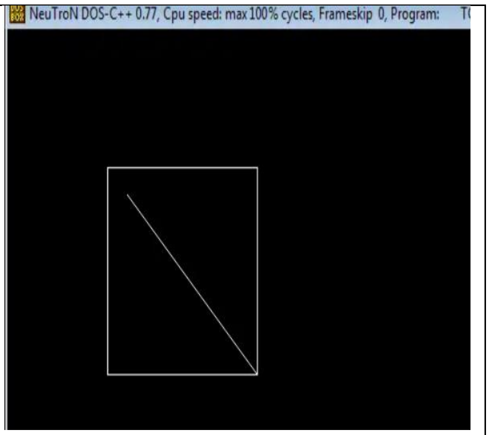
```
#include<graphics.h>

int main() {
   int gd = DETECT, gm, x, y;
   initgraph(&gd, &gm, "C:\\TC\\BGI");

   settextstyle(BOLD_FONT,HORIZ_DIR,2);
   outtextxy(220,10,"PIE CHART");
   /* Setting cordinate of center of circle */
   x = getmaxx()/2;
   y = getmaxy()/2;

   settextstyle(SANS_SERIF_FONT,HORIZ_
DIR,1);
   setfillstyle(SOLID_FILL, RED);
   pieslice(x, y, 0, 60, 120);
   outtextxy(x + 140, y - 70, "FOOD");

   setfillstyle(SOLID_FILL, YELLOW);
   pieslice(x, y, 60, 160, 120);
   outtextxy(x - 30, y - 170, "RENT");

   setfillstyle(SOLID_FILL, GREEN);
   pieslice(x, y, 160, 220, 120);
   outtextxy(x - 250, y, "ELECTRICITY");

   setfillstyle(SOLID_FILL, BROWN);
   pieslice(x, y, 220, 360, 120);
   outtextxy(x, y + 150, "SAVINGS");

   closegraph();
   return 0;
```



Exp-17 Write a program to implement Liang-Barsky 2D Line clipping.

input

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
void main(){
int i,gd=DETECT,gm;
int x1,y1,x2,y2,
xmin,xmax,ymin,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,"c:\\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 boundary");
if(q[i]>=0){if(i<2){
if(y1<ymin){y1=ymin;}
if(y2>ymax){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();}}
```

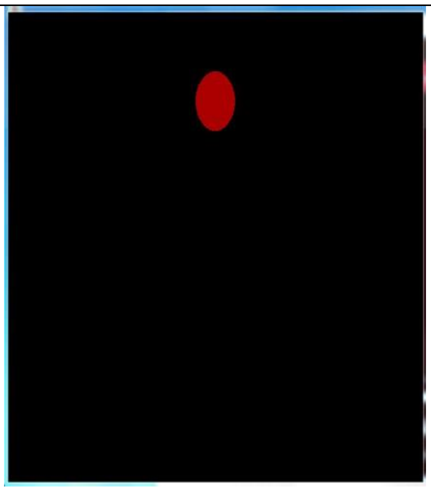

NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program: T

Exp-18 Write a program that performs a countdown for 30 seconds

```
#include <graphics.h>
#include <dos.h>
#include <conio.h>

int main()
{
   int gd = DETECT, gm, i;
   char a[5];

   initgraph( &gd, &gm, "C:\\TC\\BGI");

   settextjustify( CENTER_TEXT, CENTER_TEXT );
   settextstyle(DEFAULT_FONT,HORIZ_DIR,3);
   setcolor(RED);

   for (i = 30; i >=0; i--)
   {
      sprintf(a,"%d",i);
      outtextxy(getmaxx()/2, getmaxy()/2, a);
      delay(1000);

      if ( i == 0 )
         break;
      cleardevice();
   }

   getch();
   closegraph();
   return 0;
}
```

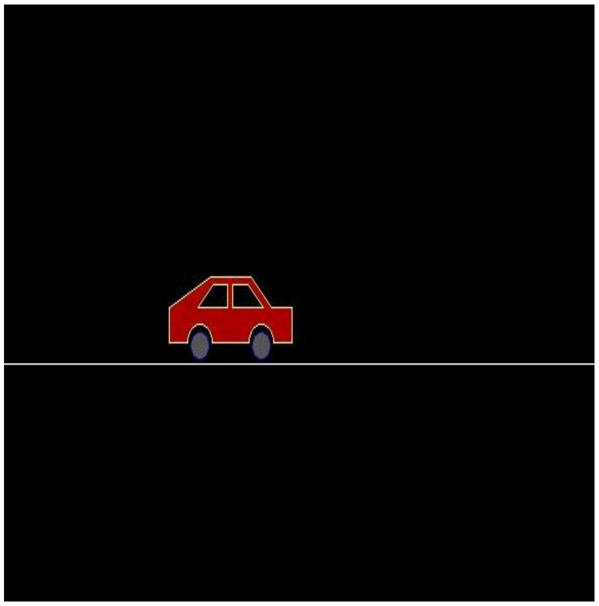
Exp-19 Write a program to implement bouncing ball animation in C/C++ using computer graphics.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main() {
        int gd = DETECT, gm = DETECT;
        int x, y = 0, j, t = 400, c = 1;
        initgraph(&gd, &gm, "");
        setcolor(RED);
        setfillstyle(SOLID_FILL, RED);
        for (x = 40; x < 602; x++) {
                cleardevice();
                circle(x, y, 30);
                floodfill(x, y, RED);
                delay(40);
                if (y >= 400) {
                        c = 0;
                        t -= 20;
                }
                if (y <= (400 - t))
                          c = 1;
                y = y + (c ? 15 : -15);
        }
        getch();
}
```



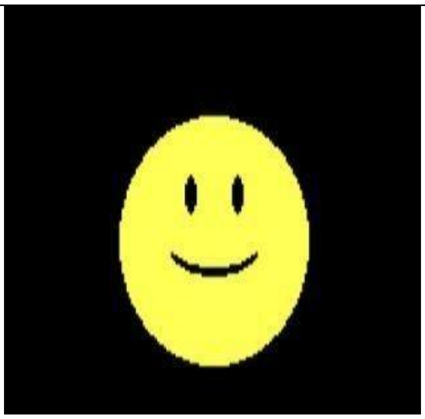
Exp-20 Write a program to implement moving car animation in C/C++ using computer graphics.

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

int main() {
   int gd = DETECT, gm;
   int i, maxx, midy;
   initgraph(&gd, &gm, "X:\\TC\\BGI");
   maxx = getmaxx();
   midy = getmaxy()/2;
   for (i=0; i < maxx-150; i=i+5) {
      cleardevice();
      setcolor(WHITE);
      line(0, midy + 37, maxx, midy + 37);
      setcolor(YELLOW);
            setfillstyle(SOLID_FILL, RED);
      line(i, midy + 23, i, midy);
      line(i, midy, 40 + i, midy - 20);
      line(40 + i, midy - 20, 80 + i, midy - 20);
      line(80 + i, midy - 20, 100 + i, midy);
      line(100 + i, midy, 120 + i, midy);
      line(120 + i, midy, 120 + i, midy + 23);
      line(0 + i, midy + 23, 18 + i, midy + 23);
      arc(30 + i, midy + 23, 0, 180, 12);
      line(42 + i, midy + 23, 78 + i, midy + 23);
      arc(90 + i, midy + 23, 0, 180, 12);
      line(102 + i, midy + 23, 120 + i, midy + 23);
      line(28 + i, midy, 43 + i, midy - 15);
      line(43 + i, midy - 15, 57 + i, midy - 15);
      line(57 + i, midy - 15, 57 + i, midy);
      line(57 + i, midy, 28 + i, midy);
      line(62 + i, midy - 15, 77 + i, midy - 15);
      line(77 + i, midy - 15, 92 + i, midy);
      line(92 + i, midy, 62 + i, midy);
      line(62 + i, midy, 62 + i, midy - 15);
      floodfill(5 + i, midy + 22, YELLOW);
      setcolor(BLUE);
      setfillstyle(SOLID_FILL, DARKGRAY);
      circle(30 + i, midy + 25, 9);
      circle(90 + i, midy + 25, 9);
      floodfill(30 + i, midy + 25, BLUE);
      floodfill(90 + i, midy + 25, BLUE);
      delay(100);
   }
   closegraph();
   return 0;
}
```



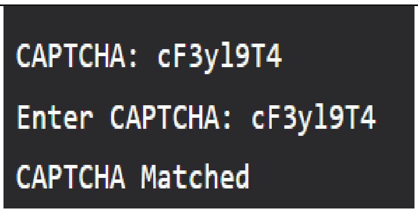
Exp-21 Write a program to draw a smiling face that appears at random positions on the screen using Computer Graphics animation

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

int main()
{

        int gr = DETECT, gm;

        initgraph(&gr, &gm, "C:\\Turboc3\\BGI");

        // Set color of smiley to yellow
        setcolor(YELLOW);

        // creating circle and fill it with
        // yellow color using floodfill.
        circle(300, 100, 40);
        setfillstyle(SOLID_FILL, YELLOW);
        floodfill(300, 100, YELLOW);

        // Set color of background to black
        setcolor(BLACK);
        setfillstyle(SOLID_FILL, BLACK);

        // Use fill ellipse for creating eyes
        fillellipse(310, 85, 2, 6);
        fillellipse(290, 85, 2, 6);

        // Use ellipse for creating mouth
        ellipse(300, 100, 205, 335, 20, 9);
        ellipse(300, 100, 205, 335, 20, 10);
        ellipse(300, 100, 205, 335, 20, 11);

        getch();
        closegraph();

        return 0;
}
```



Exp-22 Write a C program to generate a captcha which is a random string generated

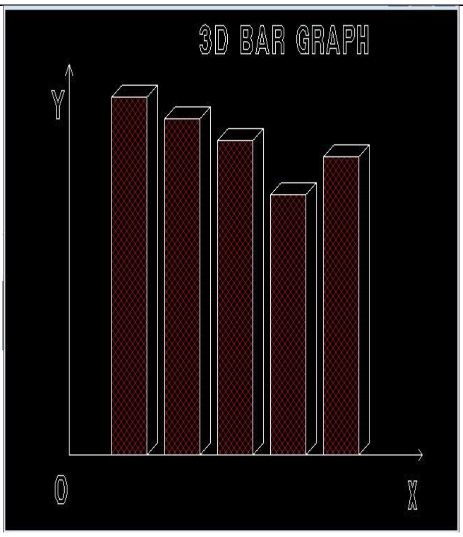
```
#include <stdlib.h>
#include <dos.h>
#include <graphics.h>
int main(){
 int i = 0, key, n, x, gd = DETECT, gm;
  char a[10];
  initgraph(&gd, &gm, "C:\\TC\\BGI");
  x = getmaxx()/2;
  settextstyle(SCRIPT_FONT, HORIZ_DIR, 5);
  settextjustify(CENTER_TEXT, CENTER_TEXT);
  setcolor(GREEN);
  outtextxy(x, 20, "CAPTCHA");
  settextstyle(SCRIPT_FONT, HORIZ_DIR, 2);
  outtextxy(x, 125, "Press any key to change the generated
\"captcha\"");
  outtextxy(x, 150, "Press escape key to exit...");

  setcolor(WHITE);
  setviewport(100, 200, 600, 400, 1);
  setcolor(RED);
  randomize();
  while (1){
    while (i < 6){
      n = random(3);
      if (n == 0)
        a[i] = 65 + random(26); /* 65 is the ASCII value of A */
      else if (n == 1)
        a[i] = 97 + random(26); /* 97 is the ASCII value of a */
      else
        a[i] = 48 + random(10); /* 48 is the ASCII value of 0 */
      i++;}
    a[i] = '\0';
    outtextxy(210, 100, a);
    key = getch();
    if (key == 27)  /* escape key */
      exit(0);
    clearviewport();
    i = 0;
  }
}
```



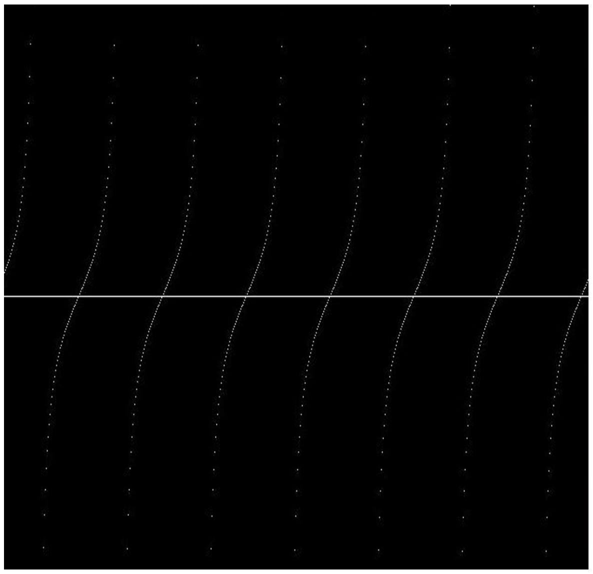
```
CAPTCHA: cF3yl9T4
Enter CAPTCHA: cF3yl9T4
CAPTCHA Matched
```

Exp-23 Write a program to draw a 3-D Bar Graph.

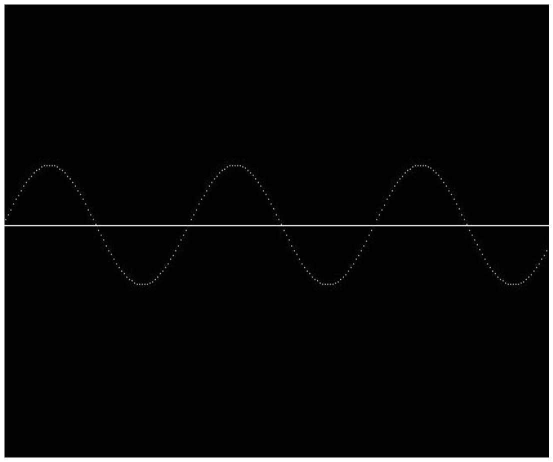
```
#include <graphics.h>

int main() {
  int gd = DETECT, gm;
  initgraph(&gd, &gm, "C:\\TC\\BGI");

  settextstyle(BOLD_FONT,HORIZ_DIR,2);
  outtextxy(275,0,"3D BAR GRAPH");

  setlinestyle(SOLID_LINE,0,2);
  /* Print X and Y Axis */
  line(90,410,90,50);
  line(90,410,590,410);
  line(85,60,90,50);
  line(95,60,90,50);
  line(585,405,590,410);
  line(585,415,590,410);

  outtextxy(65,60,"Y");
  outtextxy(570,420,"X");
  outtextxy(70,415,"O");

  /* Print 3D bars */
  setfillstyle(XHATCH_FILL, RED);
  bar3d(150,80,200,410, 15, 1);
  bar3d(225,100,275,410, 15, 1);
  bar3d(300,120,350,410, 15, 1);
  bar3d(375,170,425,410, 15, 1);
  bar3d(450,135,500,410, 15, 1);

  closegraph();
  return 0;
}
```



Exp-24 Write a C program to draw a tan graph using graphics.

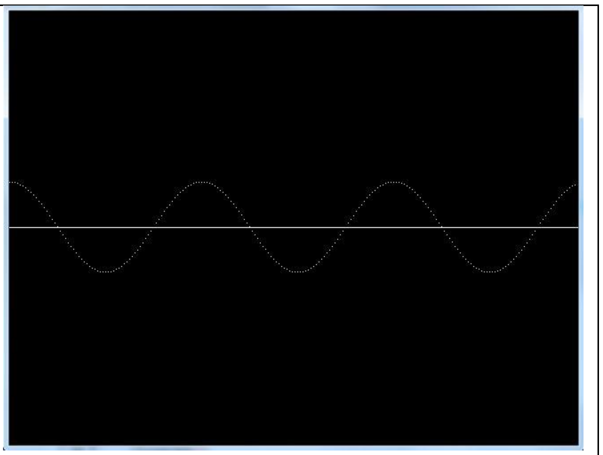
```
#include <math.h>
#include <graphics.h>
#include <dos.h>

int main() {
    int gd = DETECT, gm;
    int angle = 0;
    double x, y;

    initgraph(&gd, &gm, "C:\\TC\\BGI");

 line(0, getmaxy() / 2, getmaxx(), getmaxy() /
2);
 /* generate a sine wave */
 for(x = 0; x < getmaxx(); x++) {

     /* calculate y value given x */
     y = 50*tan(angle*3.141/180);
     y = getmaxy()/2 - y;

     /* color a pixel at the given position */
  putpixel(x, y, 15);
  delay(50);

  /* increment angle */
  angle+=2;
 }

 closegraph();

 return 0;
}
```
	

Exp-25 Write a program to draw a sine graph using C/C++ in computer graphics

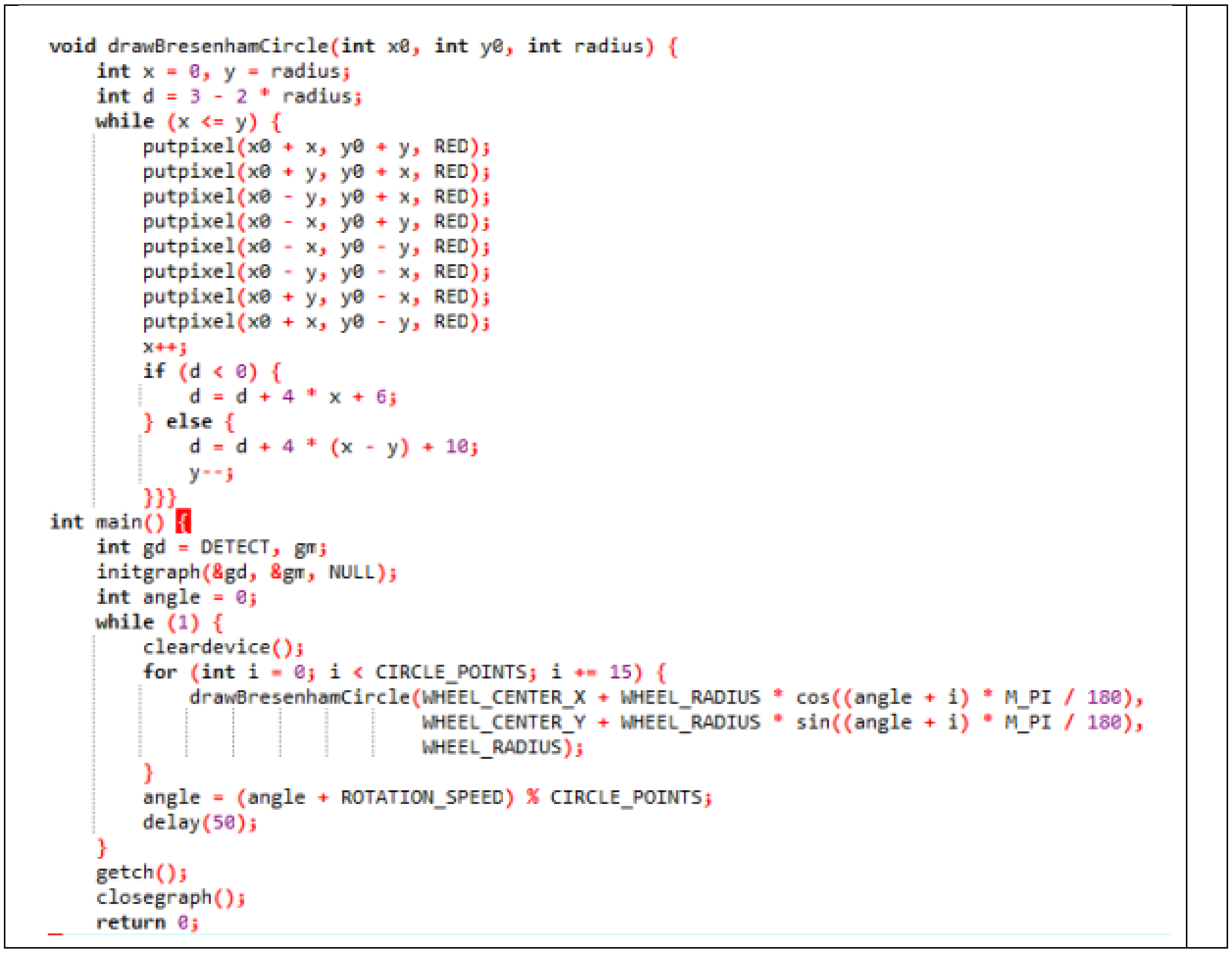
`#include <math.h>`
`#include <graphics.h>`
`#include <dos.h>`
 	
`int main() {`
`   int gd = DETECT, gm;`
`   int angle = 0;`
`   double x, y;`
`   initgraph(&gd, &gm, "C:\\TC\\BGI");`
`         line(0, getmaxy() / 2, getmaxx(),`
`getmaxy() / 2);`
`         for(x = 0; x < getmaxx(); x+=3) {`
`    y = 50*sin(angle*3.141/180);`
`    y = getmaxy()/2 - y;`
 	
`         putpixel(x, y, 15);`
`         delay(100);`
 	
`         angle+=5;`
` }`
` closegraph();`
` return 0;`
`}`
` return 0;`
`}`
	

Exp-26 Write a C program to draw a Cosine wave graph using graphics.

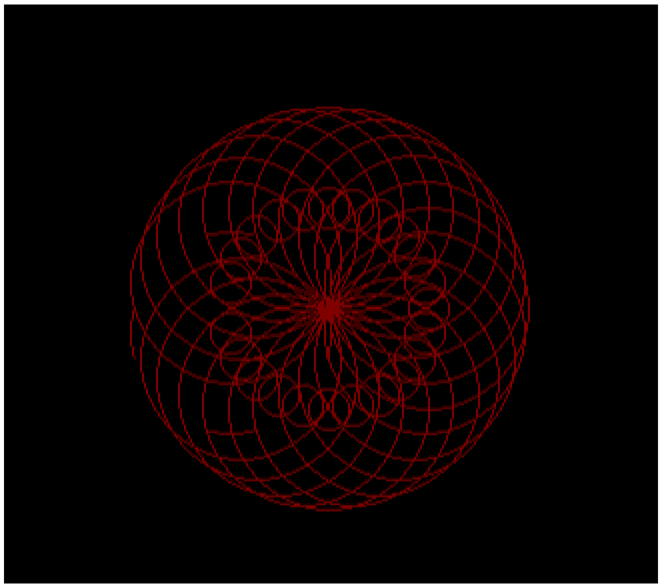
```
#include <math.h>
#include <graphics.h>
#include <dos.h>

int main() {
   int gd = DETECT, gm;
   int angle = 0;
   double x, y;

   initgraph(&gd, &gm, "C:\\TC\\BGI");

 line(0, getmaxy() / 2, getmaxx(), getmaxy() /
2);
 /* generate a sine wave */
 for(x = 0; x < getmaxx(); x+=3) {

    /* calculate y value given x */
    y = 50*sin(angle*3.141/180);
    y = getmaxy()/2 - y;

    /* color a pixel at the given position */
  putpixel(x, y, 15);
  delay(100);

  /* increment angle */
  angle+=5;
 }

 closegraph();

 return 0;
}
```



Exp-27 Write a program to generate a complete moving wheel using Midpoint circle drawing algorithm and Bresenham's circle drawing algorithm

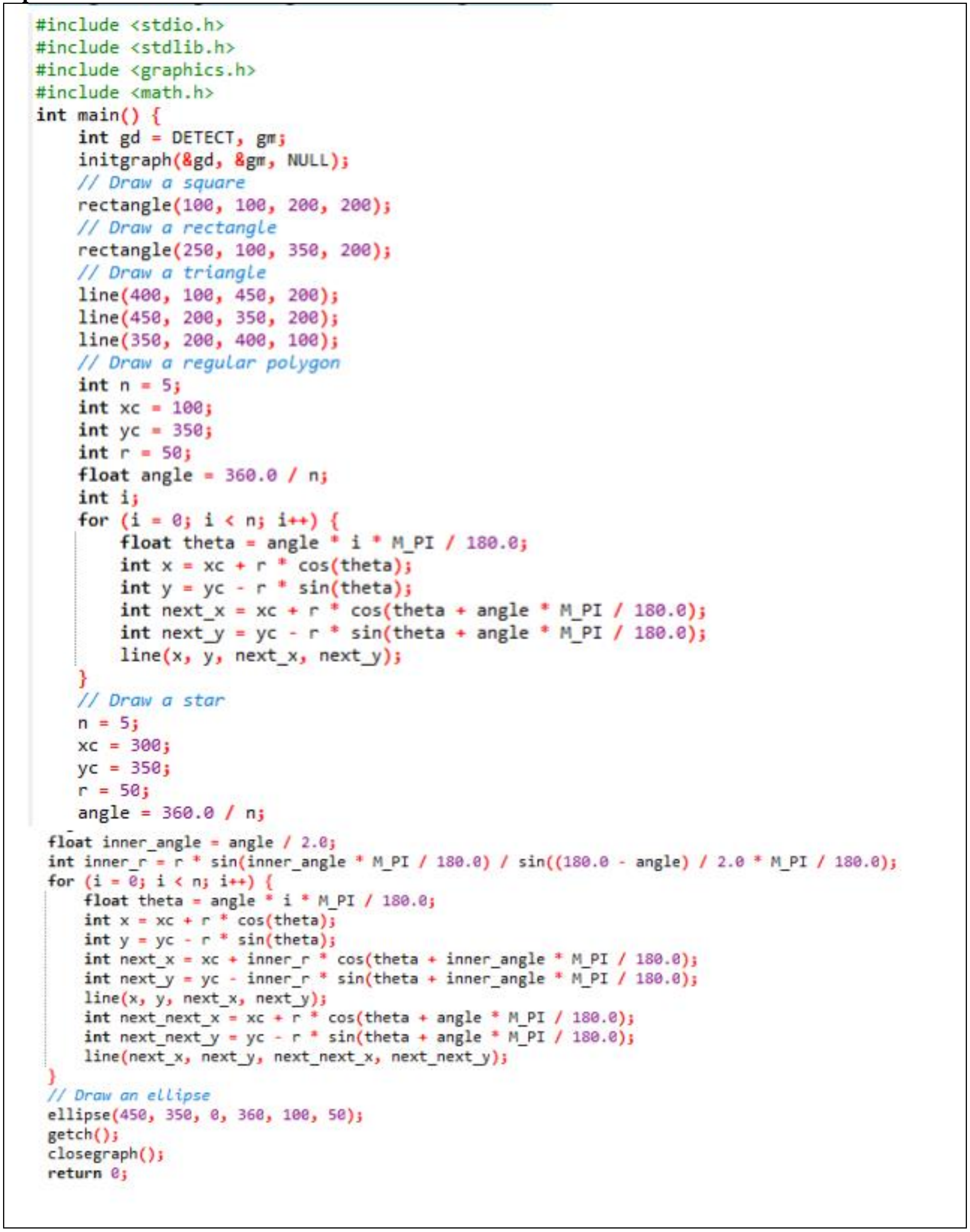


```
void drawBresenhamCircle(int x0, int y0, int radius) {
    int x = 0, y = radius;
    int d = 3 - 2 * radius;
    while (x <= y) {
        putpixel(x0 + x, y0 + y, RED);
        putpixel(x0 + y, y0 + x, RED);
        putpixel(x0 - y, y0 + x, RED);
        putpixel(x0 - x, y0 + y, RED);
        putpixel(x0 - x, y0 - y, RED);
        putpixel(x0 - y, y0 - x, RED);
        putpixel(x0 + y, y0 - x, RED);
        putpixel(x0 + x, y0 - y, RED);
        x++;
        if (d < 0) {
            d = d + 4 * x + 6;
        } else {
            d = d + 4 * (x - y) + 10;
            y--;
        }}}
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);
    int angle = 0;
    while (1) {
        cleardevice();
        for (int i = 0; i < CIRCLE_POINTS; i += 15) {
            drawBresenhamCircle(WHEEL_CENTER_X + WHEEL_RADIUS * cos((angle + i) * M_PI / 180),
                                WHEEL_CENTER_Y + WHEEL_RADIUS * sin((angle + i) * M_PI / 180),
                                WHEEL_RADIUS);
        }
        angle = (angle + ROTATION_SPEED) % CIRCLE_POINTS;
        delay(50);
    }
    getch();
    closegraph();
    return 0;
```



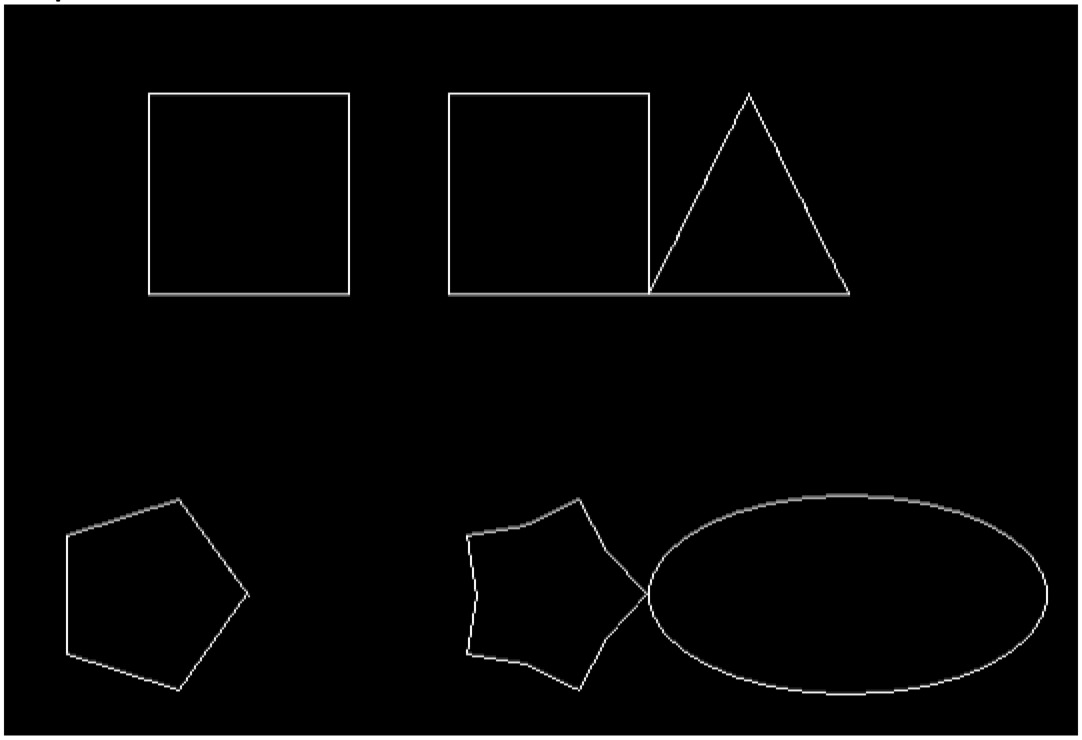
Exp 28- Write a program to draw different shapes like polygons, stars, triangles, ellipses, squares, rectangles etc

input



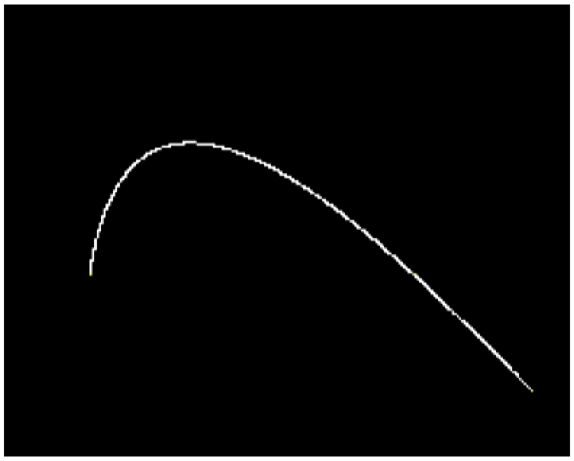
```
#include <stdio.h>
#include <stdlib.h>
#include <graphics.h>
#include <math.h>
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);
    // Draw a square
    rectangle(100, 100, 200, 200);
    // Draw a rectangle
    rectangle(250, 100, 350, 200);
    // Draw a triangle
    line(400, 100, 450, 200);
    line(450, 200, 350, 200);
    line(350, 200, 400, 100);
    // Draw a regular polygon
    int n = 5;
    int xc = 100;
    int yc = 350;
    int r = 50;
    float angle = 360.0 / n;
    int i;
    for (i = 0; i < n; i++) {
        float theta = angle * i * M_PI / 180.0;
        int x = xc + r * cos(theta);
        int y = yc - r * sin(theta);
        int next_x = xc + r * cos(theta + angle * M_PI / 180.0);
        int next_y = yc - r * sin(theta + angle * M_PI / 180.0);
        line(x, y, next_x, next_y);
    }
    // Draw a star
    n = 5;
    xc = 300;
    yc = 350;
    r = 50;
    angle = 360.0 / n;
 float inner_angle = angle / 2.0;
 int inner_r = r * sin(inner_angle * M_PI / 180.0) / sin((180.0 - angle) / 2.0 * M_PI / 180.0);
 for (i = 0; i < n; i++) {
     float theta = angle * i * M_PI / 180.0;
     int x = xc + r * cos(theta);
     int y = yc - r * sin(theta);
     int next_x = xc + inner_r * cos(theta + inner_angle * M_PI / 180.0);
     int next_y = yc - inner_r * sin(theta + inner_angle * M_PI / 180.0);
     line(x, y, next_x, next_y);
     int next_next_x = xc + r * cos(theta + angle * M_PI / 180.0);
     int next_next_y = yc - r * sin(theta + angle * M_PI / 180.0);
     line(next_x, next_y, next_next_x, next_next_y);
 }
 // Draw an ellipse
 ellipse(450, 350, 0, 360, 100, 50);
 getch();
 closegraph();
 return 0;
```

Output



Exp-29 Write a program to draw Bezier curve.

```
#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:\\tc\\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);
 }
```

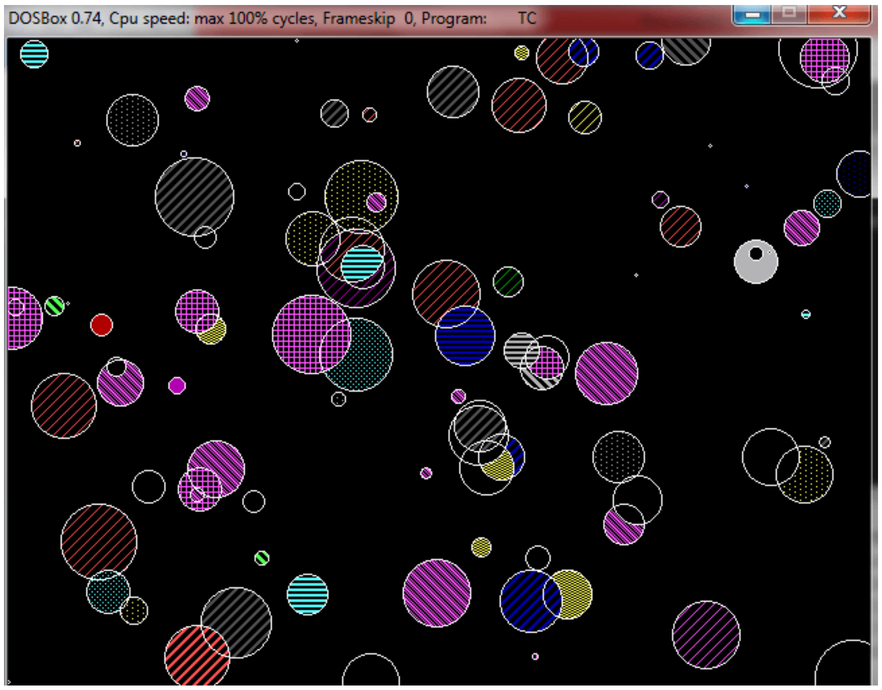


Exp-30 Program to make screen saver in that display different size circles filled with different colors

Input

```
#include<stdio.h>
#include<conio.h>
#include"graphics.h"
#include"stdlib.h"
void main()
{
   intgd=DETECT,gm,i=0,x,xx,y,yy,r;
   initgraph(&gd,&gm,"c:\\tc\\bgi");
   x=getmaxx();
   y=getmaxy();
   while(!kbhit())
   {
      i++;
      circle(xx=random(x),yy=random(y),random(30));
      setfillstyle(random(i),random(30));
      floodfill(xx,yy,getmaxcolor());
      delay(200);
   }
   getch();
}
```

Output


DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC