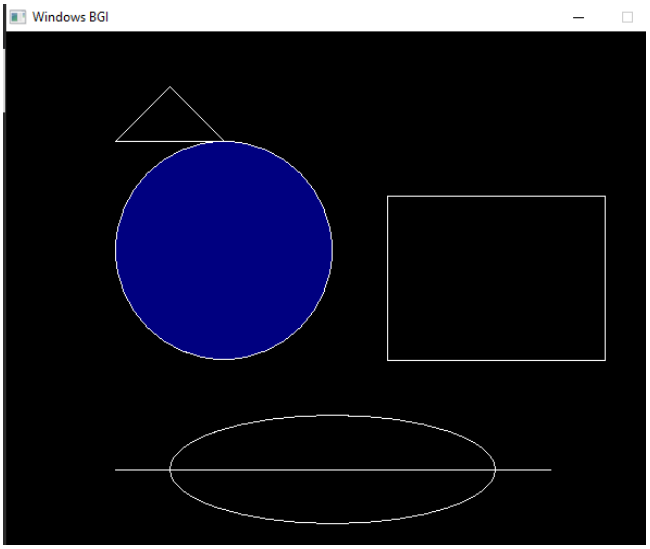
Practical: 01

Write a program to draw objects like circle , Rectangle, Polygon, etc using graphic function.

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

int main() {
   int gd = DETECT, gm;
    char data[] = "C:\\MinGW\\lib\\libbgi.a";
   initgraph(&gd, &gm, data);

   // Draw a Circle
   circle(200, 200, 100);   // (x=200, y=200, radius=100)

   // Draw a Rectangle
   rectangle(350, 150, 550, 300);  // (left, top, right, bottom)

   line(100, 400, 500, 400);  // from (100,400) to (500,400)

   ellipse(300, 400, 0, 360, 150, 50); // (x, y, start_angle, end_angle, x_radius, y_radius)

   int points[] = {100,100, 150,50, 200,100, 100,100};
   drawpoly(4, points);  // number of points (last = first)

   setfillstyle(SOLID_FILL, BLUE);
   floodfill(200, 200, WHITE); // inside circle (x,y,color_of_boundary)
   getch();
   closegraph();
   return 0;
}
```

OUTPUT

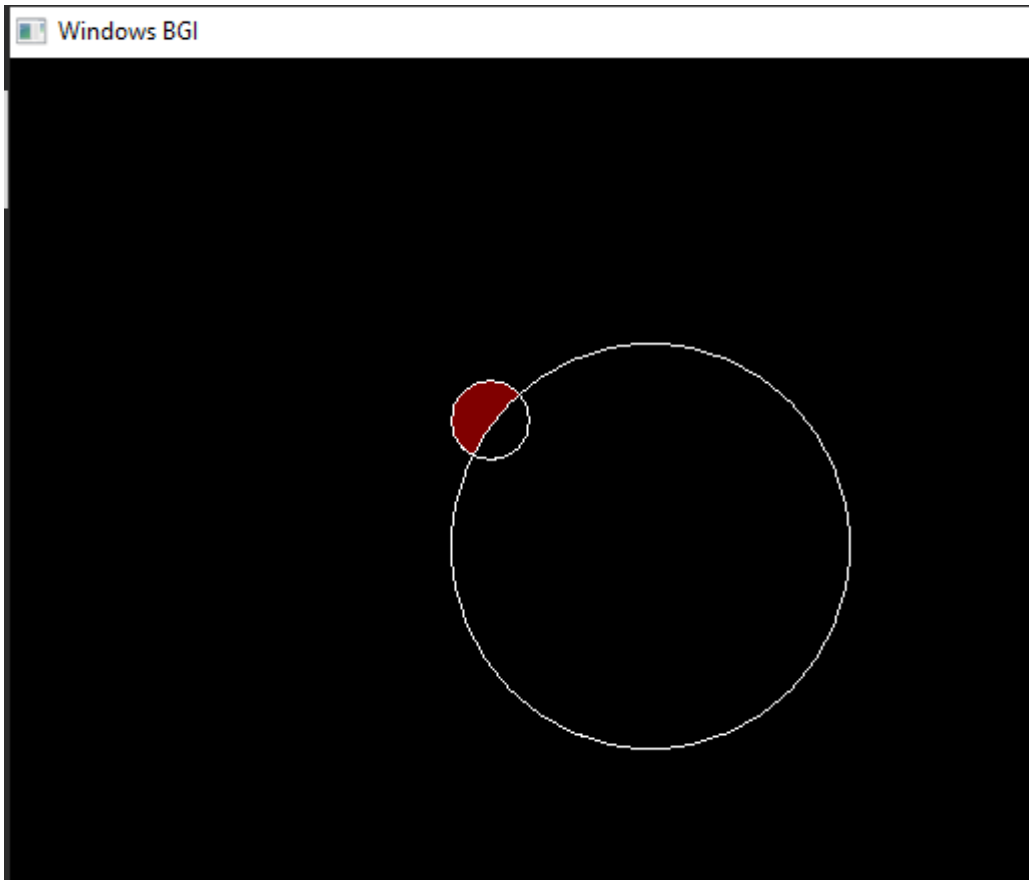

Windows BGI

Practical: 02

Write a program to rotate a circle outside another circle.

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

int main() {
   int gd = DETECT, gm;
    char data[] = "C:\\MinGW\\lib\\libbgi.a";
   initgraph(&gd, &gm, data);

   int bigX = 320, bigY = 240;   // Center of main circle
   int bigR = 100;              // Radius of main (fixed) circle
   int smallR = 20;             // Radius of rotating circle
   float angle = 0;             // Rotation angle

   while (!kbhit()) {  // keep rotating until key pressed
      cleardevice();

      // Draw fixed big circle
      circle(bigX, bigY, bigR);

      // Calculate position of small rotating circle
      int smallX = bigX + bigR * cos(angle);
      int smallY = bigY + bigR * sin(angle);

      // Draw small circle
      circle(smallX, smallY, smallR);

      // Optional: fill small circle with color
      setfillstyle(SOLID_FILL, RED);
      floodfill(smallX, smallY, WHITE);

      delay(50);     // control speed
      angle += 0.05; // increase angle
      if (angle > 2 * M_PI) angle = 0;
   }

   closegraph();
   return 0;
}
```

OUTPUT



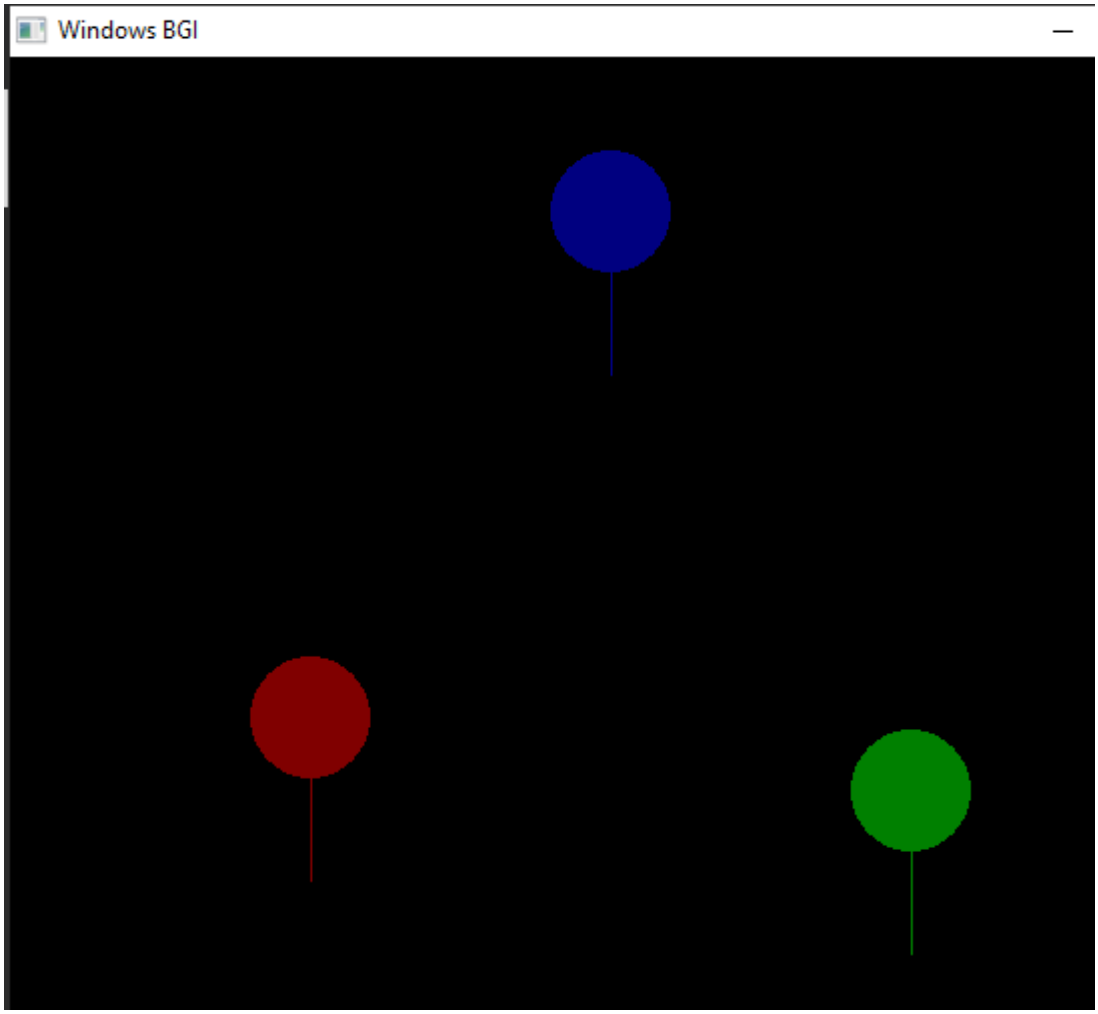
Practical: 03

Write a program to draw Flying Ballons .

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

// Function to draw a balloon
void drawBalloon(int x, int y, int color) {
   setcolor(color);
   setfillstyle(SOLID_FILL, color);

   // Balloon (circle)
   circle(x, y, 30);
   floodfill(x, y, color);

   // String
   line(x, y + 30, x, y + 80);
}

int main() {
   int gd = DETECT, gm;
    char data[] = "C:\\MinGW\\lib\\libbgi.a";
   initgraph(&gd, &gm, data);

   int y1 = 400, y2 = 450, y3 = 500;  // starting positions

   while (!kbhit()) {
      cleardevice();

      // Draw multiple balloons
      drawBalloon(150, y1, RED);
      drawBalloon(300, y2, BLUE);
      drawBalloon(450, y3, GREEN);

      // Move balloons upward
      y1 -= 2;
      y2 -= 3;
      y3 -= 4;

      // Reset balloons when they go out of screen
      if (y1 < 0) y1 = getmaxy() + 50;
      if (y2 < 0) y2 = getmaxy() + 50;
```

```
        if (y3 < 0) y3 = getmaxy() + 50;

        delay(50);
    }

    closegraph();
    return 0;
}
```

OUTPUT



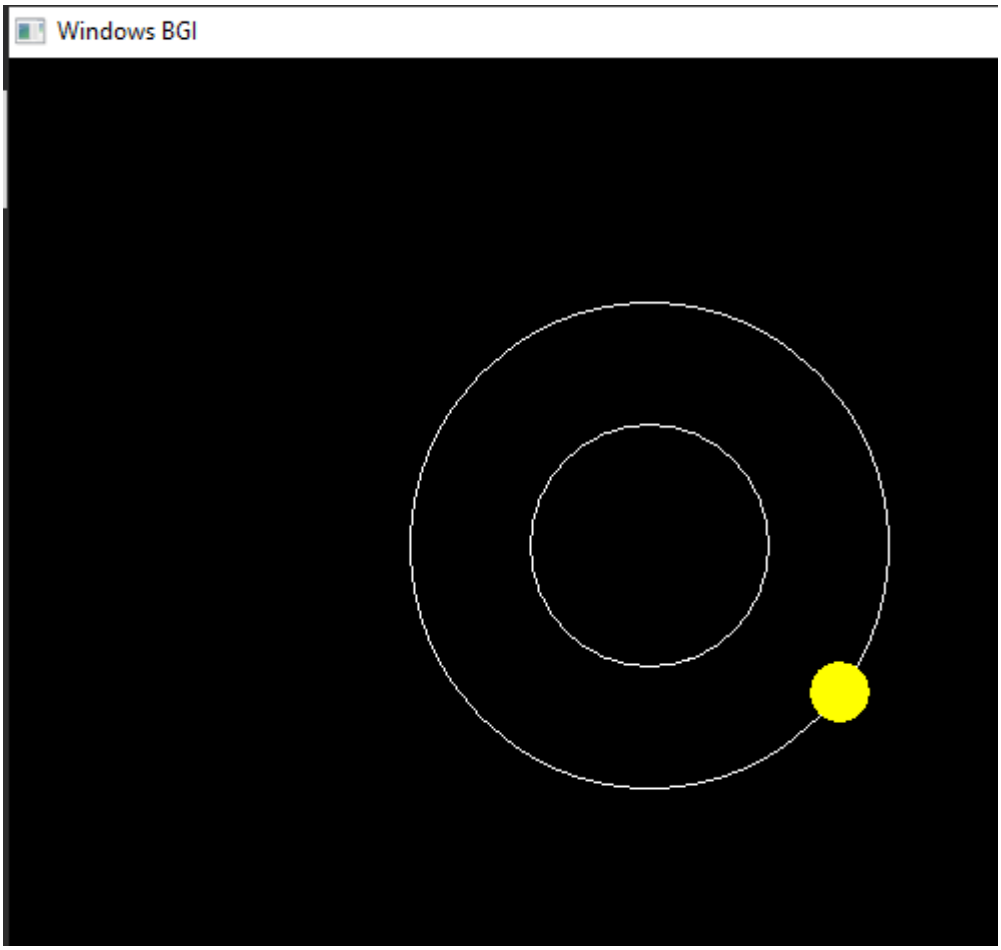
Practical: 04

Write a program to rotate circle inside and outside another circle alternatively.

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

int main() {
   int gd = DETECT, gm;
   char data[] = "C:\\MinGW\\lib\\libbgi.a";
   initgraph(&gd, &gm, data);

   int bigX = 320, bigY = 240;   // Center of main circle
   int bigR = 120;              // Outer radius
   int innerR = 60;             // Inner radius
   int smallR = 15;             // Rotating circle radius
   float angle = 0;             // Rotation angle
   bool outside = true;         // flag for outside/inside mode

   while (!kbhit()) {
      cleardevice();

      // Draw reference circles (main + inner orbit)
      setcolor(WHITE);
      circle(bigX, bigY, bigR);   // outer circle
      circle(bigX, bigY, innerR); // inner circle

      // Decide which orbit radius to use
      int orbitR = outside ? bigR : innerR;

      // Calculate small circle position
      int smallX = bigX + orbitR * cos(angle);
      int smallY = bigY + orbitR * sin(angle);

      // Draw rotating circle
      setcolor(YELLOW);
      setfillstyle(SOLID_FILL, YELLOW);
      circle(smallX, smallY, smallR);
      floodfill(smallX, smallY, YELLOW);

      delay(50);
```

```
        // update angle
        angle += 0.05;
        if (angle > 2 * M_PI) {
            angle = 0;
            outside = !outside;  // switch between outside & inside orbit
        }
    }

    closegraph();
    return 0;
}
```

OUTPUT



Practical: 05

Write a program to draw a Hut .

```
#include <graphics.h>

#include <conio.h>

int main() {
intgd = DETECT, gm;
char data[] = “C:\\MinGW\\lib\\libbgi.a”;
initgraph(&gd, &gm, data);
setbkcolor(BLACK);
cleardevice();
setcolor(WHITE_PEN);
 Rectangle(150, 250, 350, 400);
 Rectangle(230, 300, 270, 400);
Line(150, 250, 250, 150); // Left side of roof
 Line(250, 150, 350, 250); // Right side of roof
 Rectangle(170, 270, 200, 300); // Left window
 Rectangle(300, 270, 330, 300); // Right window

getch();
closegraph();
return 0;

}
```

OUTPUT

Practical: 06

Write a program for DDA line drawing algorithm .

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

int main() {
   float x0, y0, x1, y1;
printf("Enter starting point (x0 y0): ");
scanf("%f %f", &x0, &y0);
printf("Enter ending point (x1 y1): ");
scanf("%f %f", &x1, &y1);

   float dx = x1 - x0;
   float dy = y1 - y0;

int steps = (fabs(dx) >fabs(dy)) ? fabs(dx) : fabs(dy);

   float Xinc = dx / steps;
   float Yinc = dy / steps;

   float x = x0;
   float y = y0;

printf("\nPoints on the line:\n");
   for(inti = 0; i<= steps; i++) {
printf("(%.2f, %.2f)\n", x, y);
      x += Xinc;
      y += Yinc;
   }

   return 0;
}
```

OUTPUT

```
Enter starting point (x0 y0): 1 2
Enter ending point (x1 y1): 4 5

Points on the line:
(1.00, 2.00)
(2.00, 3.00)
(3.00, 4.00)
(4.00, 5.00)
```

Practical: 07

Write a program for Bresenham’sline drawing algorithm .

```
#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <iostream>
using namespace std;

void bresenhamLine(int x1, int y1, int x2, int y2) {
   int dx = abs(x2 - x1);
   intdy = abs(y2 - y1);

   intsx = (x1 < x2) ? 1 : -1;  // step for x
   intsy = (y1 < y2) ? 1 : -1;  // step for y

   int err = dx - dy;

   while (true) {
      putpixel(x1, y1, WHITE);  // draw pixel at (x1,y1)

      if (x1 == x2 && y1 == y2) break;

      int e2 = 2 * err;

      if (e2 > -dy) {
         err -= dy;
         x1 += sx;
      }
      if (e2 < dx) {
         err += dx;
         y1 += sy;
      }
   }
}

int main() {
   intgd = DETECT, gm;
    char data[] = "C:\\MinGW\\lib\\libbgi.a";
   initgraph(&gd, &gm, data);

   int x1, y1, x2, y2;
   cout<< "Enter x1 y1: ";
   cin>> x1 >> y1;
```

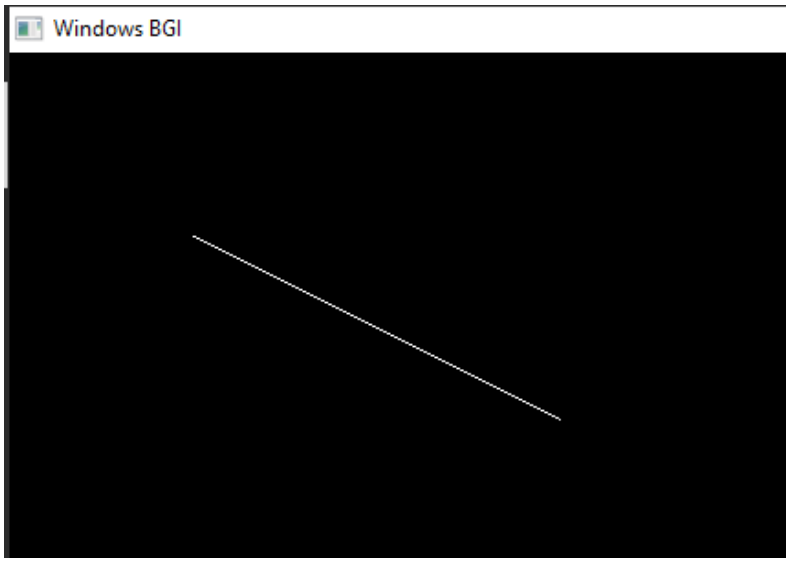
```
    cout<< "Enter x2 y2: ";
    cin>> x2 >> y2;

    bresenhamLine(x1, y1, x2, y2);

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

OUTPUT

- **Inputs**

```
Enter x1 y1: 1200 1300
Enter x2 y2: 1400 1600
PS E:\graphics.h-project-template-main>
PS E:\graphics.h-project-template-main> & "e:\graphics.h-p
c\bresenham.exe"
Enter x1 y1: 100 500
Enter x2 y2: 400 800
PS E:\graphics.h-project-template-main>
```

- Line Drawn



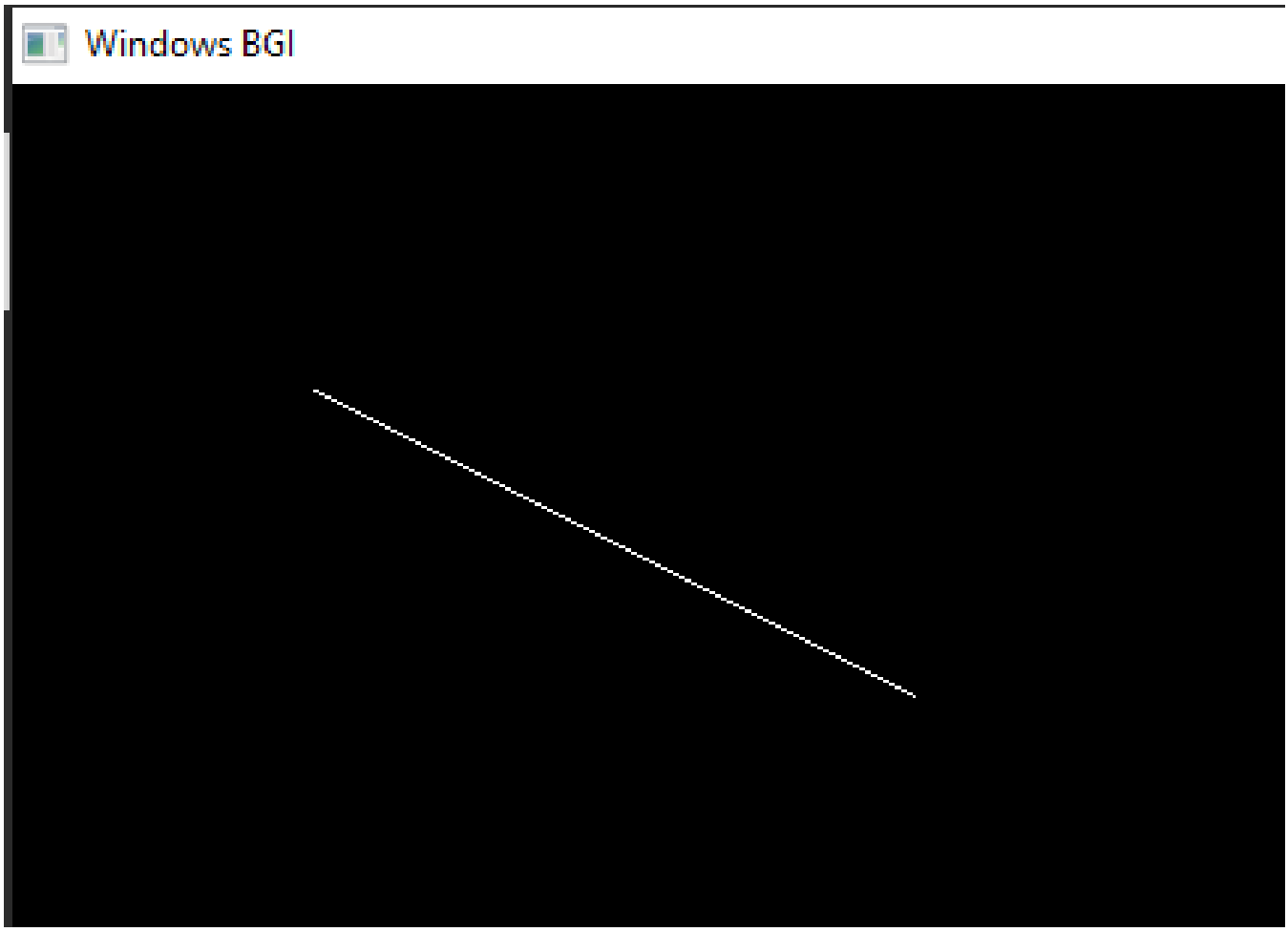
Practical: 08

Write a program for Mid point line algorithm .

```
#include <iostream>
#include <graphics.h>
using namespace std;

void midPointLine(int x1, int y1, int x2, int y2) {
   int dx = x2 - x1;
   intdy = y2 - y1;

   int d = 2 * dy - dx;
   intincrE = 2 * dy;
   intincrNE = 2 * (dy - dx);

   int x = x1, y = y1;

   putpixel(x, y, WHITE); // draw first point

   while (x < x2) {
      if (d <= 0) {
         d += incrE;
         x++;
      } else {
         d += incrNE;
         x++;
         y++;
      }
      putpixel(x, y, WHITE); // plot next point
      delay(20); // small delay for visualization
   }
}

int main() {
   intgd = DETECT, gm;
   char data[] = "C:\\MinGW\\lib\\libbgi.a";
   initgraph(&gd, &gm, data);



   // Example: pre-defined points
   int x1 = 100, y1 = 100;
   int x2 = 300, y2 = 200;

   midPointLine(x1, y1, x2, y2);
```

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

OUTPUT



Practical: 09

Write a program for Bresenham's Circle drawing algorithm .

```
#include <iostream>
#include <graphics.h>
using namespace std;

void drawCirclePoints(int xc, intyc, int x, int y) {
   putpixel(xc + x, yc + y, WHITE);
   putpixel(xc - x, yc + y, WHITE);
   putpixel(xc + x, yc - y, WHITE);
   putpixel(xc - x, yc - y, WHITE);
   putpixel(xc + y, yc + x, WHITE);
   putpixel(xc - y, yc + x, WHITE);
   putpixel(xc + y, yc - x, WHITE);
   putpixel(xc - y, yc - x, WHITE);
}

void bresenhamCircle(int xc, intyc, int r) {
   int x = 0;
   int y = r;
   int d = 3 - 2 * r;

   while (x <= y) {
      drawCirclePoints(xc, yc, x, y);

      if (d < 0) {
         d = d + 4 * x + 6;
      } else {
         d = d + 4 * (x - y) + 10;
         y--;
      }
      x++;
      delay(20); // just for visualization
   }
}

int main() {
   intgd = DETECT, gm;
    char data[] = "C:\\MinGW\\lib\\libbgi.a";
   initgraph(&gd, &gm, data);

   // Predefined circle center and radius
   int xc = 200, yc = 200, r = 100;
```

```
    bresenhamCircle(xc, yc, r);

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

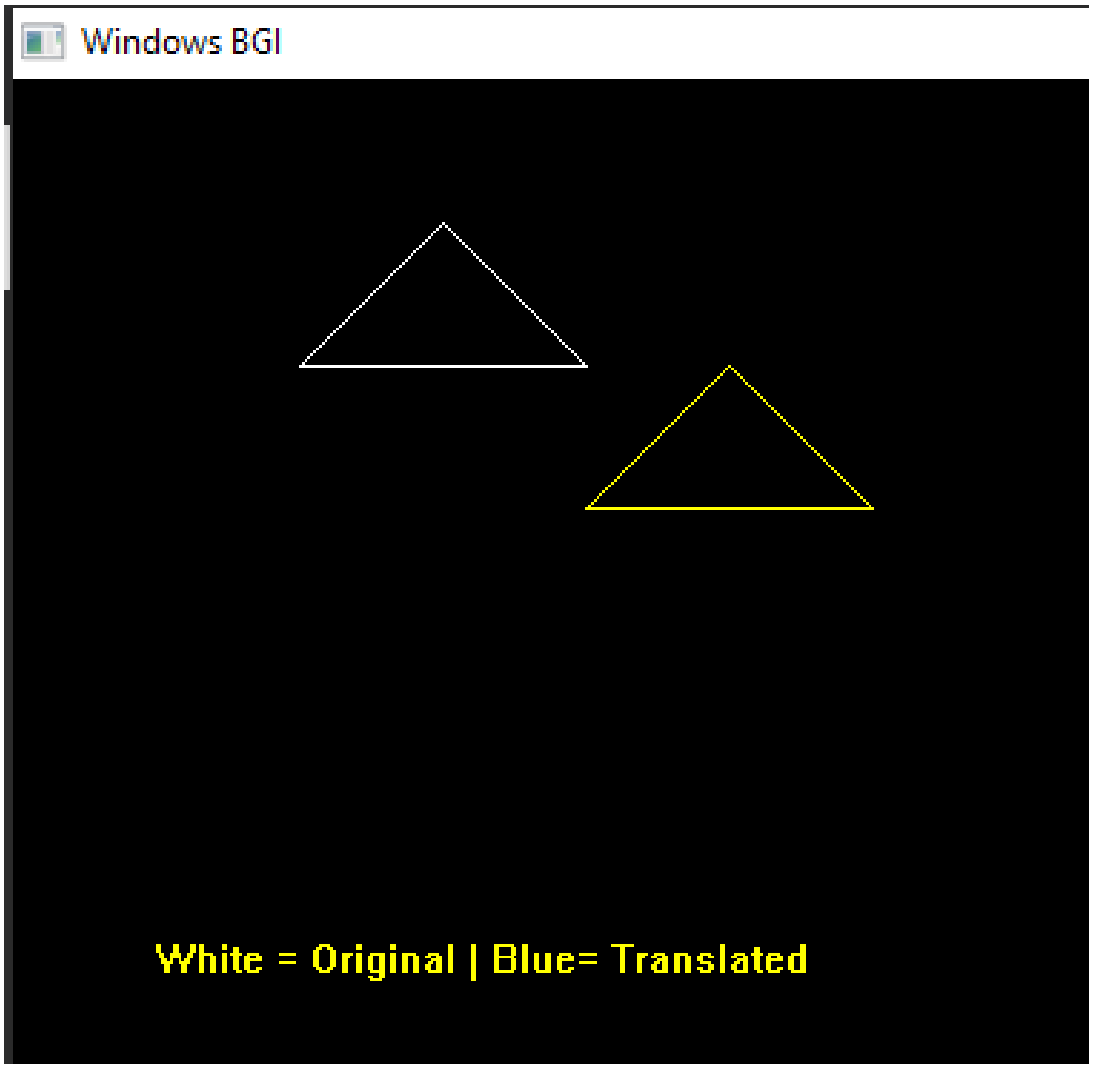
OUTPUT

Practical: 10

Write a program for Translation in 2D.

```
#include <iostream>
#include <graphics.h>
using namespace std;

int main() {
   intgd = DETECT, gm;
    char data[] = "C:\\MinGW\\lib\\libbgi.a";
   initgraph(&gd, &gm, data);

   // Define a triangle
   int x1 = 100, y1 = 100;
   int x2 = 200, y2 = 100;
   int x3 = 150, y3 = 50;

   // Draw original triangle
   setcolor(WHITE);
   line(x1, y1, x2, y2);
   line(x2, y2, x3, y3);
   line(x3, y3, x1, y1);

   // Translation factors
   inttx = 100, ty = 50;

   // Apply translation
   int x1t = x1 + tx, y1t = y1 + ty;
   int x2t = x2 + tx, y2t = y2 + ty;
   int x3t = x3 + tx, y3t = y3 + ty;

   // Draw translated triangle
   setcolor(YELLOW);
   line(x1t, y1t, x2t, y2t);
   line(x2t, y2t, x3t, y3t);
   line(x3t, y3t, x1t, y1t);
   // Show translation vector
   setcolor(YELLOW);
   outtextxy(50, 300, "White = Original | Blue= Translated");

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

OUTPUT

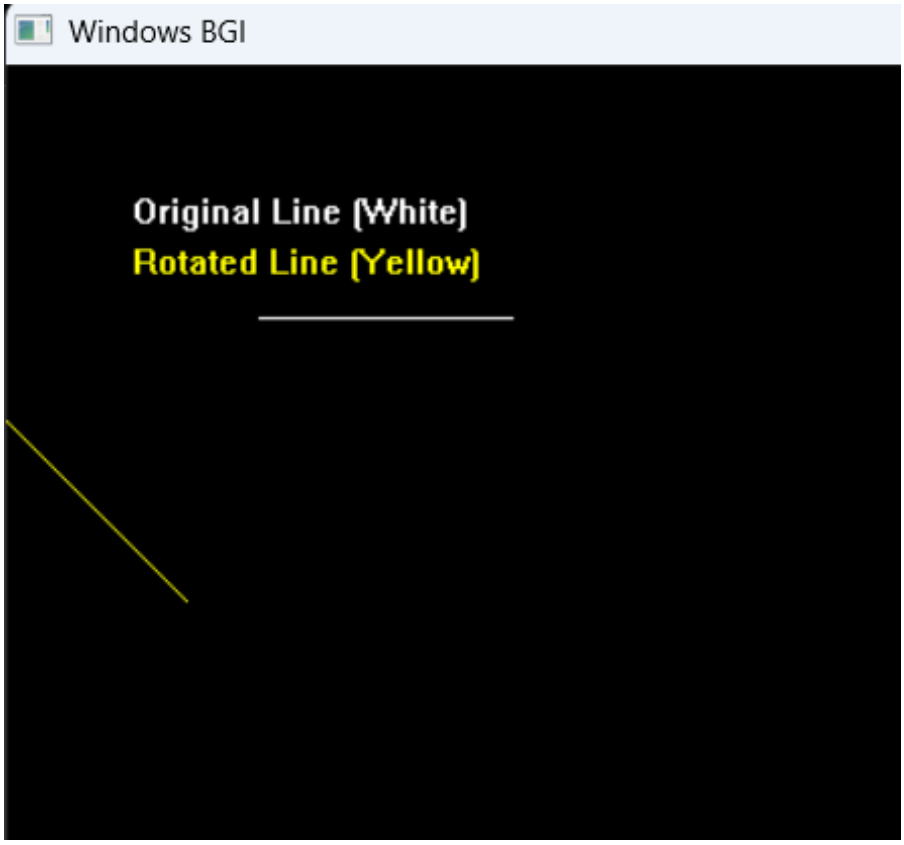


Practical: 11

Write a program for Rotation in 2D.

```
#include <graphics.h>
#include <conio.h>
#include <math.h>
#include <iostream>
using namespace std;

int main()
{
   int gd = DETECT, gm;
   char data[] = "C:\\MinGW\\lib\\libbgi.a";
   initgraph(&gd, &gm, data);

   // Original line coordinates
   int x1 = 100, y1 = 100;
   int x2 = 200, y2 = 100;

   // Rotation angle in degrees
   float angle;
   cout << "Enter rotation angle (in degrees): ";
   cin >> angle;
   // Convert to radians
   float rad = angle * (3.14159 / 180.0);
   // Rotation formula
   int x1r = round(x1 * cos(rad) - y1 * sin(rad));
   int y1r = round(x1 * sin(rad) + y1 * cos(rad));
   int x2r = round(x2 * cos(rad) - y2 * sin(rad));
   int y2r = round(x2 * sin(rad) + y2 * cos(rad));
   // Draw original line
   setcolor(WHITE);
   line(x1, y1, x2, y2);
   outtextxy(50, 50, (char*)"Original Line (White)");
   // Draw rotated line
   setcolor(RED);
   line(x1r, y1r, x2r, y2r);
   outtextxy(50, 70, (char*)"Rotated Line (Red)");
   getch();
   closegraph();
   return 0;
}
```

OUTPUT



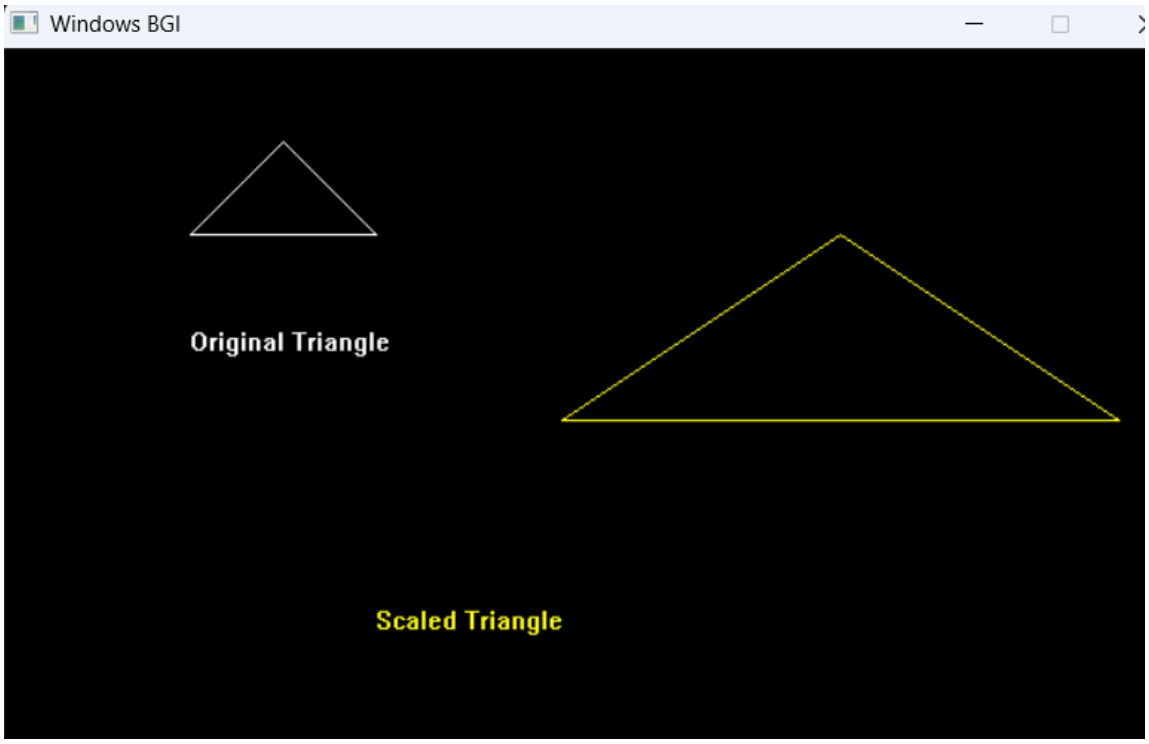
Practical: 12

Write a program for Scaling in 2D.

```
#include <graphics.h>
#include <conio.h>
#include <iostream>
using namespace std;

int main() {
   int gd = DETECT, gm;
    char data[] = "C:\\MinGW\\lib\\libbgi.a";
   initgraph(&gd, &gm, data);

   // Original coordinates of a triangle
   int x1 = 100, y1 = 100;
   int x2 = 200, y2 = 100;
   int x3 = 150, y3 = 50;

   // Draw original triangle
   setcolor(WHITE);
   line(x1, y1, x2, y2);
   line(x2, y2, x3, y3);
   line(x3, y3, x1, y1);
   outtextxy(100, 150, "Original Triangle");

   // Scaling factors
   float sx, sy;
   cout << "Enter scaling factor Sx: ";
   cin >> sx;
   cout << "Enter scaling factor Sy: ";
   cin >> sy;

   // Reference point (pivot) for scaling (origin)
   int x_ref = 0, y_ref = 0;

   // Apply scaling transformation
   int x1_new = x_ref + (x1 - x_ref) * sx;
   int y1_new = y_ref + (y1 - y_ref) * sy;
   int x2_new = x_ref + (x2 - x_ref) * sx;
   int y2_new = y_ref + (y2 - y_ref) * sy;
   int x3_new = x_ref + (x3 - x_ref) * sx;
   int y3_new = y_ref + (y3 - y_ref) * sy;

   // Draw scaled triangle
```

```
    setcolor(YELLOW);
    line(x1_new, y1_new, x2_new, y2_new);
    line(x2_new, y2_new, x3_new, y3_new);
    line(x3_new, y3_new, x1_new, y1_new);
    outtextxy(200, 300, "Scaled Triangle");

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

OUTPUT



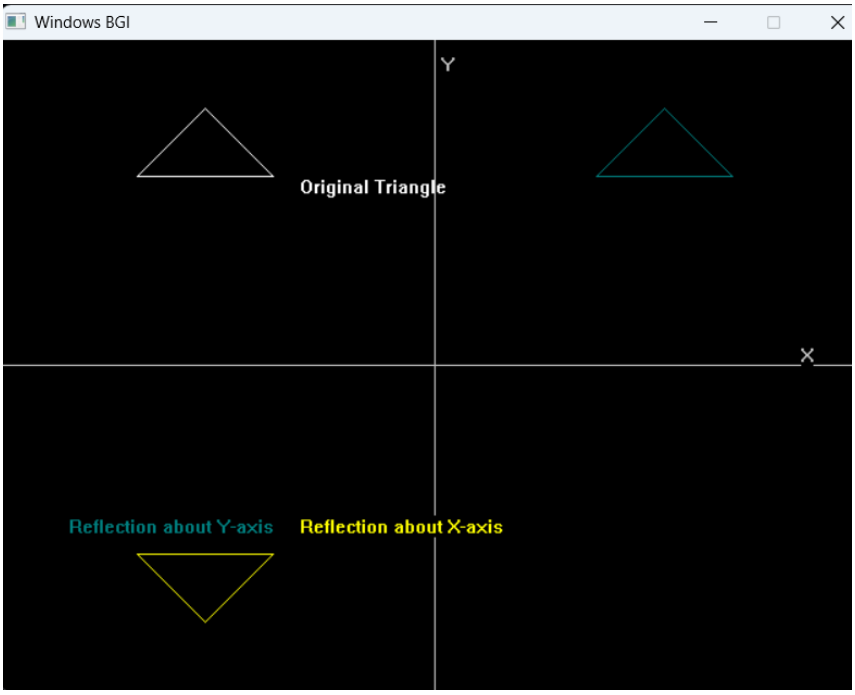
Practical: 13

Write a program for Reflection in 2D.

```
#include <graphics.h>
#include <conio.h>
#include <iostream>
using namespace std;

int main() {
   int gd = DETECT, gm;
   char data[] = "C:\\MinGW\\lib\\libbgi.a";
   initgraph(&gd, &gm, data);

   // Original coordinates of the triangle
   int x1 = 100, y1 = 100;
   int x2 = 150, y2 = 50;
   int x3 = 200, y3 = 100;

   // Draw original triangle
   setcolor(WHITE);
   line(x1, y1, x2, y2);
   line(x2, y2, x3, y3);
   line(x3, y3, x1, y1);
   outtextxy(220, 100, "Original Triangle");

   // Draw X and Y axes
   setcolor(LIGHTGRAY);
   line(0, getmaxy()/2, getmaxx(), getmaxy()/2); // X-axis
   line(getmaxx()/2, 0, getmaxx()/2, getmaxy()); // Y-axis
   outtextxy(getmaxx()/2 + 5, 10, "Y");
   outtextxy(getmaxx() - 50, getmaxy()/2 - 15, "X");

   // Shift origin to center of screen
   int midx = getmaxx()/2;
   int midy = getmaxy()/2;

   int X1 = x1 - midx, Y1 = midy - y1;
   int X2 = x2 - midx, Y2 = midy - y2;
   int X3 = x3 - midx, Y3 = midy - y3;

   // -------- Reflection about X-axis --------
   int X1x = X1, Y1x = -Y1;
   int X2x = X2, Y2x = -Y2;
   int X3x = X3, Y3x = -Y3;
```

```
    X1x += midx; Y1x = midy - Y1x;
    X2x += midx; Y2x = midy - Y2x;
    X3x += midx; Y3x = midy - Y3x;

    setcolor(YELLOW);
    line(X1x, Y1x, X2x, Y2x);
    line(X2x, Y2x, X3x, Y3x);
    line(X3x, Y3x, X1x, Y1x);
    outtextxy(220, 350, "Reflection about X-axis");
    int X1y = -X1, Y1y = Y1;
    int X2y = -X2, Y2y = Y2;
    int X3y = -X3, Y3y = Y3;

    X1y += midx; Y1y = midy - Y1y;
    X2y += midx; Y2y = midy - Y2y;
    X3y += midx; Y3y = midy - Y3y;

    setcolor(CYAN);
    line(X1y, Y1y, X2y, Y2y);
    line(X2y, Y2y, X3y, Y3y);
    line(X3y, Y3y, X1y, Y1y);
    outtextxy(50, 350, "Reflection about Y-axis");

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

OUTPUT



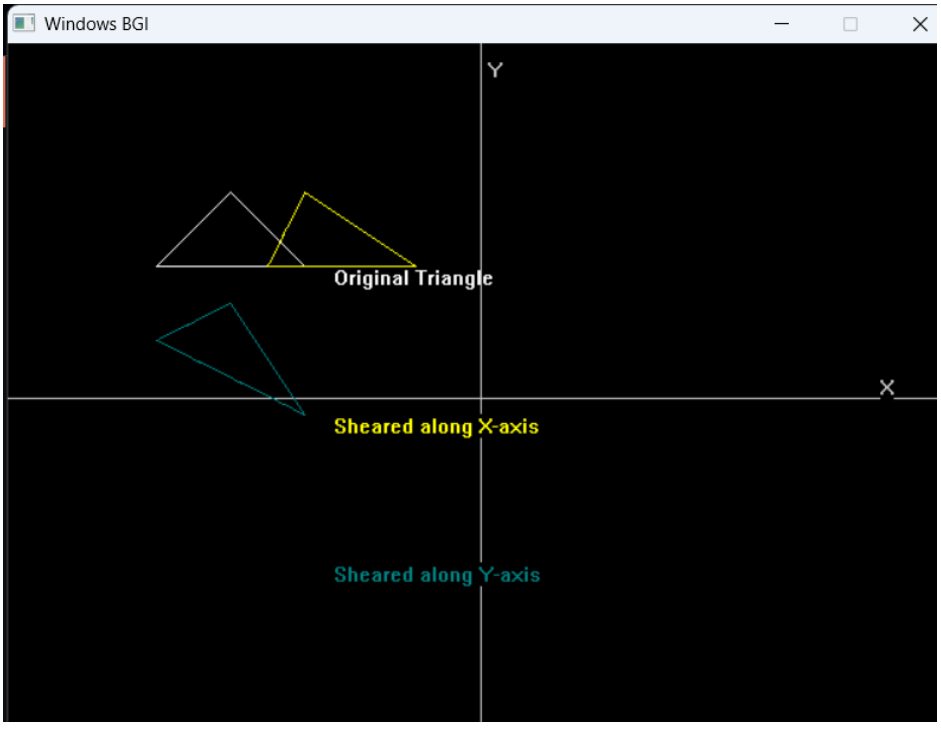
Practical: 14

Write a program for Shearing in 2D.

```
#include <graphics.h>
#include <conio.h>
#include <iostream>
using namespace std;

int main() {
   int gd = DETECT, gm;
   char data[] = "C:\\MinGW\\lib\\libbgi.a";
   initgraph(&gd, &gm, data);

   // Original triangle coordinates
   int x1 = 100, y1 = 150;
   int x2 = 200, y2 = 150;
   int x3 = 150, y3 = 100;

   // Draw original triangle
   setcolor(WHITE);
   line(x1, y1, x2, y2);
   line(x2, y2, x3, y3);
   line(x3, y3, x1, y1);
   outtextxy(220, 150, "Original Triangle");

   // Draw X and Y axes
   setcolor(LIGHTGRAY);
   line(0, getmaxy()/2, getmaxx(), getmaxy()/2); // X-axis
   line(getmaxx()/2, 0, getmaxx()/2, getmaxy()); // Y-axis
   outtextxy(getmaxx()/2 + 5, 10, "Y");
   outtextxy(getmaxx() - 50, getmaxy()/2 - 15, "X");

   // Shearing factors
   float shx = 0.5;  // Shear along X-axis
   float shy = 0.5;  // Shear along Y-axis

   // ---------- Shearing along X-axis ----------
   int x1x = x1 + y1 * shx;
   int y1x = y1;
   int x2x = x2 + y2 * shx;
   int y2x = y2;
   int x3x = x3 + y3 * shx;
   int y3x = y3;
```

```
    setcolor(YELLOW);
    line(x1x, y1x, x2x, y2x);
    line(x2x, y2x, x3x, y3x);
    line(x3x, y3x, x1x, y1x);
    outtextxy(220, 250, "Sheared along X-axis");

    // ---------- Shearing along Y-axis ----------
    int x1y = x1;
    int y1y = y1 + x1 * shy;
    int x2y = x2;
    int y2y = y2 + x2 * shy;
    int x3y = x3;
    int y3y = y3 + x3 * shy;

    setcolor(CYAN);
    line(x1y, y1y, x2y, y2y);
    line(x2y, y2y, x3y, y3y);
    line(x3y, y3y, x1y, y1y);
    outtextxy(220, 350, "Sheared along Y-axis");

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

OUTPUT



Practical: 15

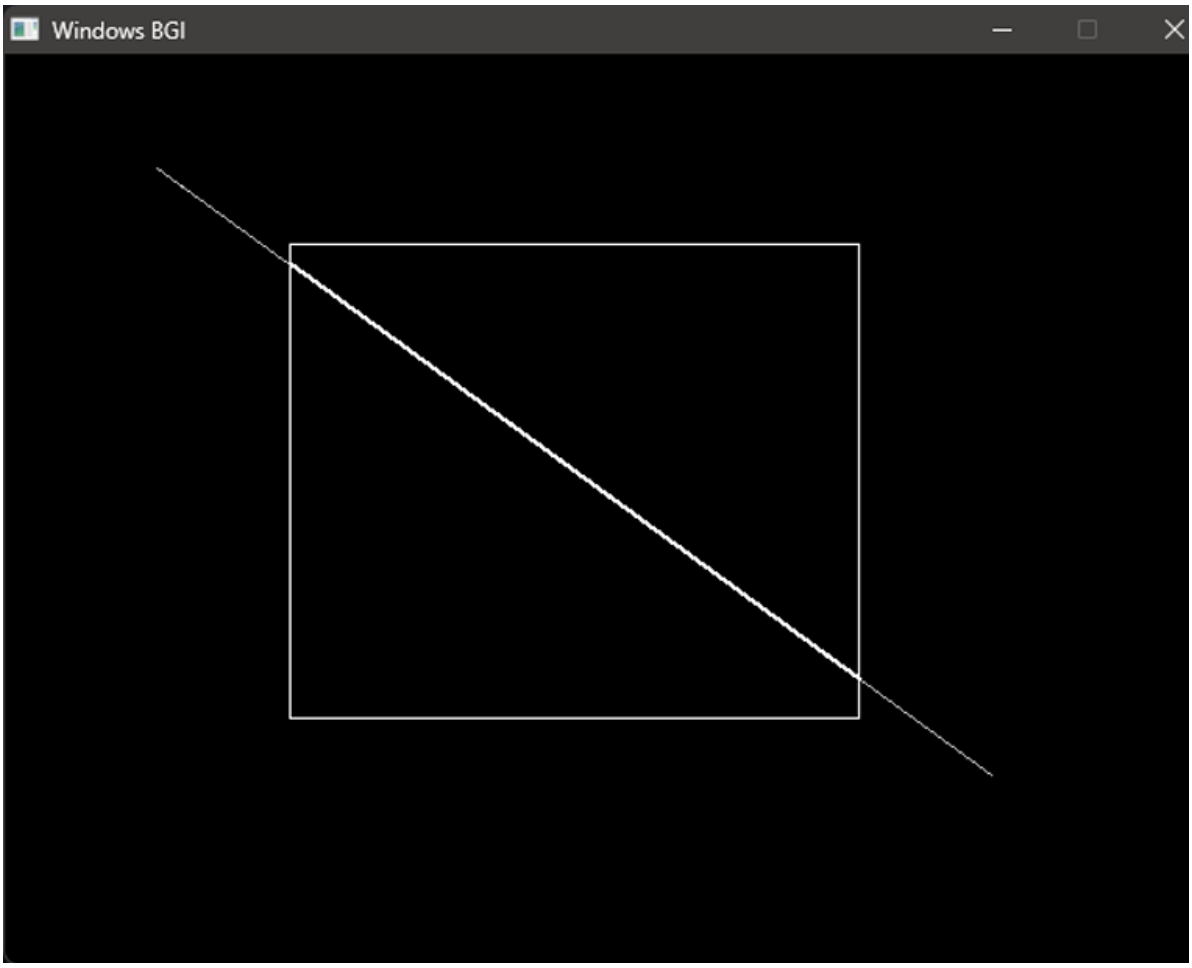
Write a program to implement Cohen Sutherland's Algorithm.

```
#include <graphics.h>
// Region codes for Cohen-Sutherland
const int INSIDE = 0;  // 0000
const int LEFT   = 1;  // 0001
const int RIGHT  = 2;  // 0010
const int BOTTOM = 4;  // 0100
const int TOP    = 8;  // 1000
int computeCode(int x, int y, int xmin, int ymin, int xmax, int ymax) {
  int code = INSIDE;
  if (x < xmin) code |= LEFT;
  else if (x > xmax) code |= RIGHT;
  if (y < ymin) code |= TOP;     // Note: in BGI, y increases downward; window
defined with ymin as top
  else if (y > ymax) code |= BOTTOM;
  return code;
}
bool cohenSutherlandClip(int &x0, int &y0, int &x1, int &y1, int xmin, int ymin,
int xmax, int ymax) {
  int code0 = computeCode(x0, y0, xmin, ymin, xmax, ymax);
  int code1 = computeCode(x1, y1, xmin, ymin, xmax, ymax);

  bool accept = false;
  while (true) {
    if ((code0 | code1) == 0) { // both inside
      accept = true;
      break;
    } else if (code0 & code1) { // both share an outside zone -> reject
      break;
    } else { // some segment of line is within the window
      int outCode = code0 ? code0 : code1;
      int x, y;
      // Avoid division by zero by handling vertical and horizontal specially
      if (outCode & TOP) { // point is above window (y < ymin)
        x = x0 == x1 ? x0 : x0 + (x1 - x0) * (ymin - y0) / (y1 - y0);
        y = ymin;
      } else if (outCode & BOTTOM) { // point is below window (y > ymax)
        x = x0 == x1 ? x0 : x0 + (x1 - x0) * (ymax - y0) / (y1 - y0);
        y = ymax;
      } else if (outCode & RIGHT) { // point is to the right of window
        y = y0 == y1 ? y0 : y0 + (y1 - y0) * (xmax - x0) / (x1 - x0);
        x = xmax;
```

```
    } else { // LEFT
      y = y0 == y1 ? y0 : y0 + (y1 - y0) * (xmin - x0) / (x1 - x0);
      x = xmin;
    }

    if (outCode == code0) {
      x0 = x; y0 = y;
      code0 = computeCode(x0, y0, xmin, ymin, xmax, ymax);
    } else {
      x1 = x; y1 = y;
      code1 = computeCode(x1, y1, xmin, ymin, xmax, ymax);
    }
   }
  }

  return accept;
}

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

  // Define clipping window (top-left to bottom-right)
  int xmin = 150, ymin = 100, xmax = 450, ymax = 350;

  // Example line endpoints (intentionally crossing the window)
  int x0 = 80, y0 = 60;
  int x1 = 520, y1 = 380;

  // Draw clipping window
  setcolor(WHITE);
  rectangle(xmin, ymin, xmax, ymax);
  // Draw original line
  setcolor(RED);
  line(x0, y0, x1, y1);
  // Perform clipping
  int cx0 = x0, cy0 = y0, cx1 = x1, cy1 = y1;
  if (cohenSutherlandClip(cx0, cy0, cx1, cy1, xmin, ymin, xmax, ymax)) {
   setcolor(GREEN);
   setlinestyle(SOLID_LINE, 0, 2);
   line(cx0, cy0, cx1, cy1);
  } else {
   setcolor(LIGHTGRAY);
   outtextxy(10, 10, (char*)"Line completely outside window");
  }
```

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

OUTPUT

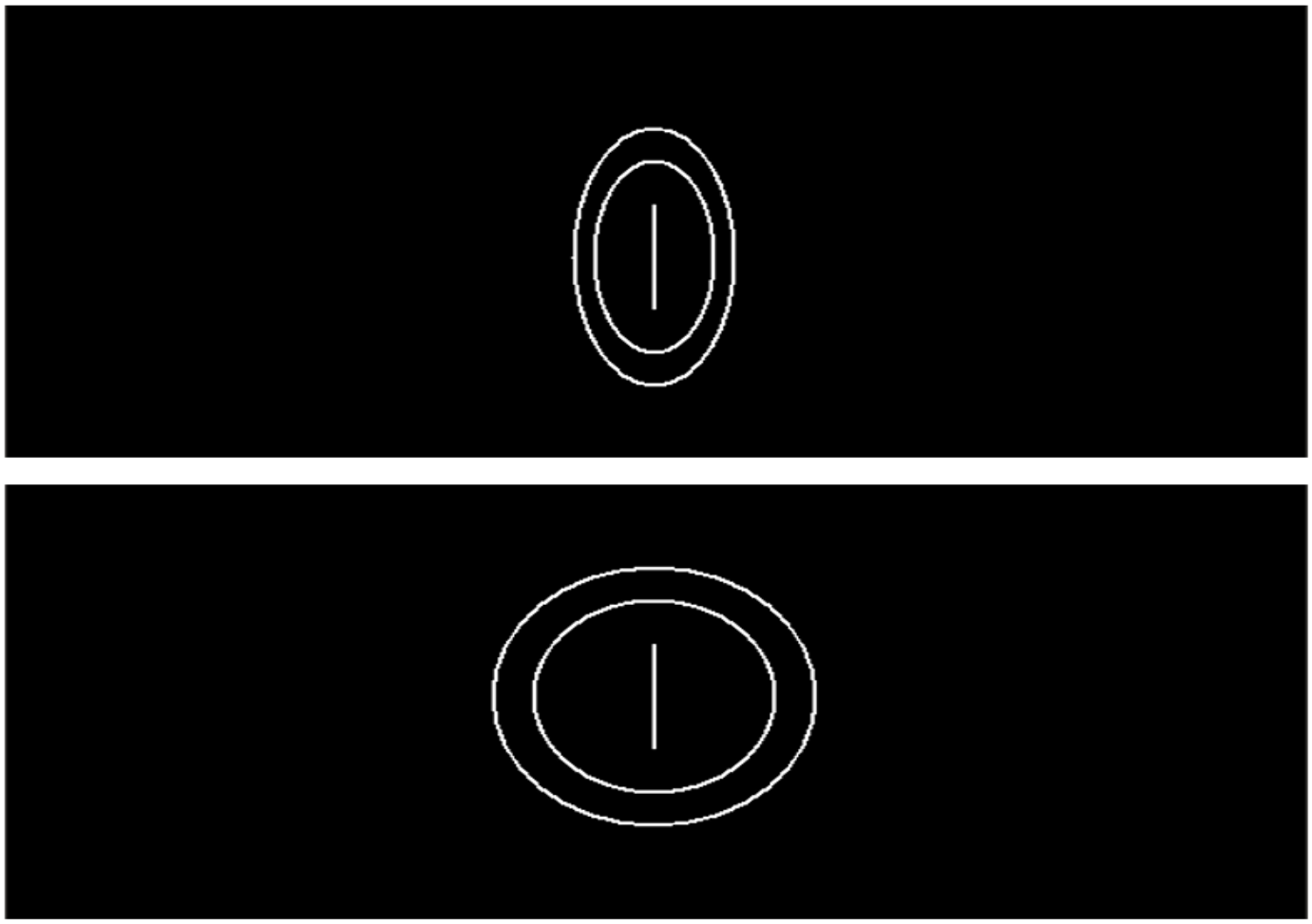


Practical: 16

Write a program to rotate a coin .

```
#include <graphics.h>
#include <math.h>
#include <conio.h>
int main() {
  int gd = DETECT, gm;
  initgraph(&gd, &gm, NULL);

  setbkcolor(BLACK);
  cleardevice();
  int cx = getmaxx() / 2;
  int cy = getmaxy() / 2;
  int r = (getmaxx() < getmaxy() ? getmaxx() : getmaxy()) / 6; // base radius

  setcolor(WHITE);
  setlinestyle(SOLID_LINE, 0, 2);

  // Animate until a key is pressed
  while (!kbhit()) {
    for (int deg = 0; deg < 360 && !kbhit(); deg += 6) {
      cleardevice();
      // Simulate rotation by compressing the x-radius with cos(theta)
      double theta = deg * 3.1415926535 / 180.0;
      int rx = (int)(fabs(cos(theta)) * r) + 1; // +1 to avoid disappearing entirely
      int ry = r;

      // Outer edge of the coin
      ellipse(cx, cy, 0, 360, rx, ry);
      // Inner ring to give a coin-like look (optional outline only, no fill)
      ellipse(cx, cy, 0, 360, (int)(rx * 0.75), (int)(ry * 0.75));
      // A simple "head" mark on the coin: a vertical line that foreshortens
      // Use scaled length based on rx/ry to suggest perspective
      int markLen = (int)(ry * 0.4);
      line(cx, cy - markLen, cx, cy + markLen);
      delay(25);
    }
  getch();
  closegraph();
  return 0;
}
```

OUTPUT



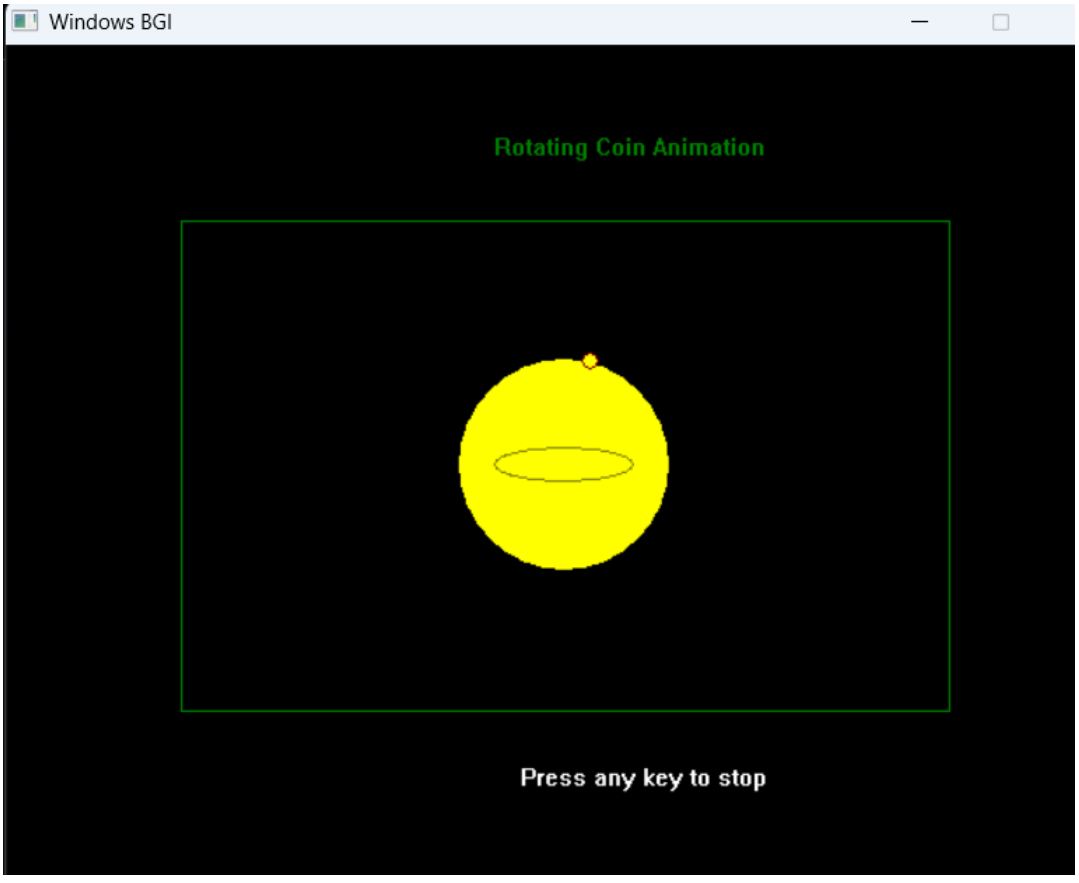
Practical: 17

Write a program to rotate a coin on table.

```
#include <graphics.h>
#include <conio.h>
#include <cmath>
#include <iostream>
using namespace std;

int main() {
   int gd = DETECT, gm;
   char data[] = "C:\\MinGW\\lib\\libbgi.a";
   initgraph(&gd, &gm, data);

   int midx = getmaxx() / 2;   // center of screen (table center)
   int midy = getmaxy() / 2;
   int radius = 60;           // radius of coin

   setbkcolor(BLACK);       // background color (table)
   cleardevice();

   // Draw static table border
   setcolor(GREEN);
   rectangle(100, 100, getmaxx() - 100, getmaxy() - 100);
   outtextxy(midx - 40, 50, "Rotating Coin Animation");

   float angle = 0;  // rotation angle in radians

   // Animation loop
   while (!kbhit()) {
      cleardevice();

      // Draw table
      setcolor(GREEN);
      rectangle(100, 100, getmaxx() - 100, getmaxy() - 100);
      outtextxy(midx - 40, 50, "Rotating Coin Animation");
      setcolor(WHITE);
      outtextxy(midx - 25, getmaxy() - 70, "Press any key to stop");

      // Draw coin outer circle
      setcolor(YELLOW);
      circle(midx, midy, radius);
      setfillstyle(SOLID_FILL, YELLOW);
      floodfill(midx, midy, YELLOW);
```

```
    // Draw inner design (rotating ellipse to show spin)
    setcolor(BROWN);
    int rx = 40;   // horizontal radius of ellipse
    int ry = 40 * abs(cos(angle)); // changes with rotation to simulate thickness
    ellipse(midx, midy, 0, 360, rx, ry);

    // Small decorative star to indicate rotation direction
    int dotx = midx + radius * cos(angle);
    int doty = midy + radius * sin(angle);
    setcolor(RED);
    fillellipse(dotx, doty, 5, 5);

    delay(30);    // control speed
    angle += 0.1; // rotate angle
  }

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

OUTPUT



Practical: 18

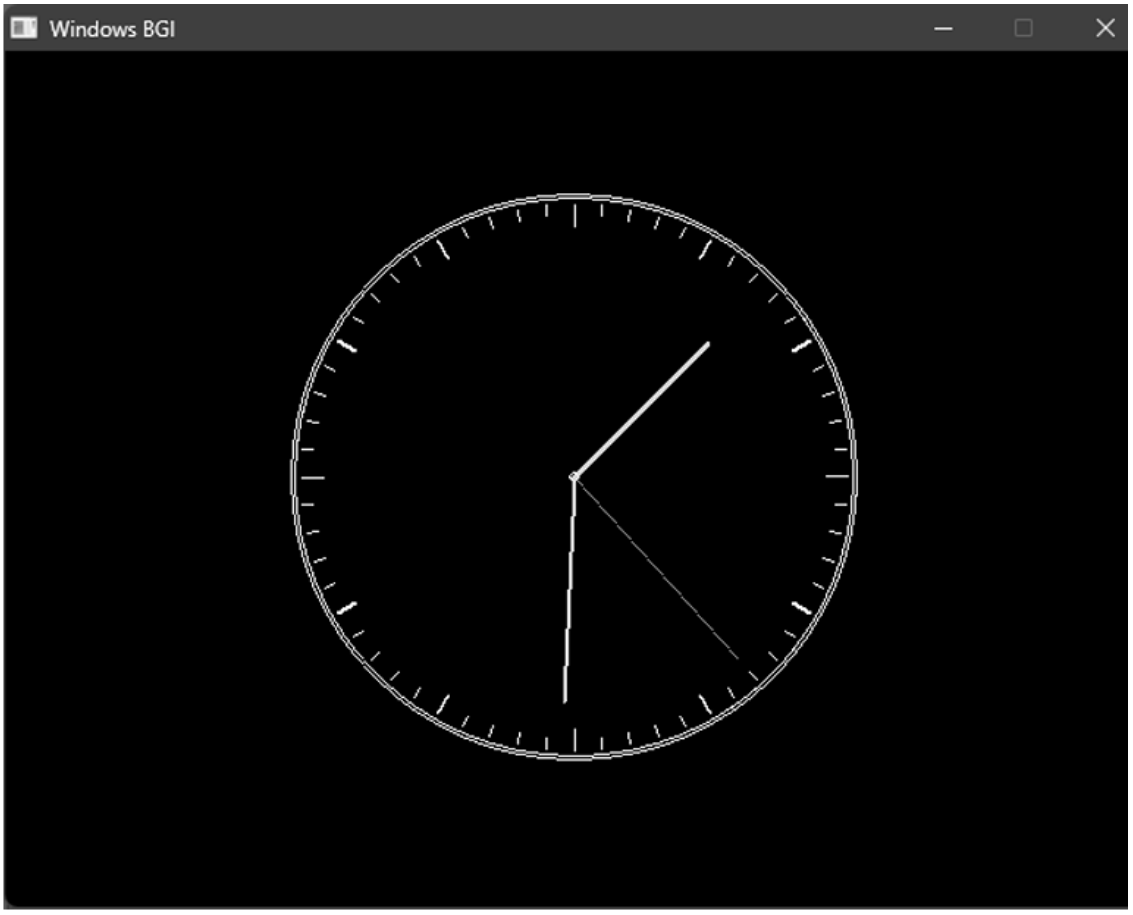
Write a program for showing working of Analog Clock.

```
#include <graphics.h>
#include <math.h>
#include <time.h>
#include <conio.h>

static const double PI_D = 3.14159265358979323846;

void drawClockFace(int cx, int cy, int r) {
  setcolor(WHITE);
  setlinestyle(SOLID_LINE, 0, 1);
  circle(cx, cy, r);
  circle(cx, cy, r - 2);

  // Minute ticks (60)
  for (int m = 0; m < 60; m++) {
    double a = 2 * PI_D * (m / 60.0);
    int x1 = (int)(cx + (r - 6) * sin(a));
    int y1 = (int)(cy - (r - 6) * cos(a));
    int x2 = (int)(cx + (r - 12) * sin(a));
    int y2 = (int)(cy - (r - 12) * cos(a));
    line(x1, y1, x2, y2);
  }

  // Hour ticks (12) - thicker/longer
  setlinestyle(SOLID_LINE, 0, 2);
  for (int h = 0; h < 12; h++) {
    double a = 2 * PI_D * (h / 12.0);
    int x1 = (int)(cx + (r - 6) * sin(a));
    int y1 = (int)(cy - (r - 6) * cos(a));
    int x2 = (int)(cx + (r - 18) * sin(a));
    int y2 = (int)(cy - (r - 18) * cos(a));
    line(x1, y1, x2, y2);
  }
}
int main() {
  int gd = DETECT, gm;
  initgraph(&gd, &gm, NULL);

  setbkcolor(BLACK);
  cleardevice();
  int cx = getmaxx() / 2;
```

```
int cy = getmaxy() / 2;
int r = (getmaxx() < getmaxy() ? getmaxx() : getmaxy()) / 3;

// Run until key press
while (!kbhit()) {
  // Current time
  time_t now = time(NULL);
  struct tm *lt = localtime(&now);
  int hh = lt->tm_hour % 12;
  int mm = lt->tm_min;
  int ss = lt->tm_sec;

  // Redraw face
  cleardevice();
  drawClockFace(cx, cy, r);

  // Angles: measured from 12 o'clock, clockwise
  double secAngle = 2 * PI_D * (ss / 60.0);
  double minAngle = 2 * PI_D * ((mm + ss / 60.0) / 60.0);
  double hourAngle = 2 * PI_D * ((hh + mm / 60.0 + ss / 3600.0) / 12.0);

  // Second hand
  setcolor(LIGHTRED);
  setlinestyle(SOLID_LINE, 0, 1);
  int sx = (int)(cx + (r - 22) * sin(secAngle));
  int sy = (int)(cy - (r - 22) * cos(secAngle));
  line(cx, cy, sx, sy);

  // Minute hand
  setcolor(LIGHTCYAN);
  setlinestyle(SOLID_LINE, 0, 2);
  int mx = (int)(cx + (r - 32) * sin(minAngle));
  int my = (int)(cy - (r - 32) * cos(minAngle));
  line(cx, cy, mx, my);

  // Hour hand
  setcolor(LIGHTGREEN);
  setlinestyle(SOLID_LINE, 0, 3);
  int hx = (int)(cx + (r - 52) * sin(hourAngle));
  int hy = (int)(cy - (r - 52) * cos(hourAngle));
  line(cx, cy, hx, hy);

  // Center cap
  setcolor(WHITE);
  setlinestyle(SOLID_LINE, 0, 1);
```

```
    circle(cx, cy, 3);

    delay(100); // refresh rate
  }

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

OUTPUT



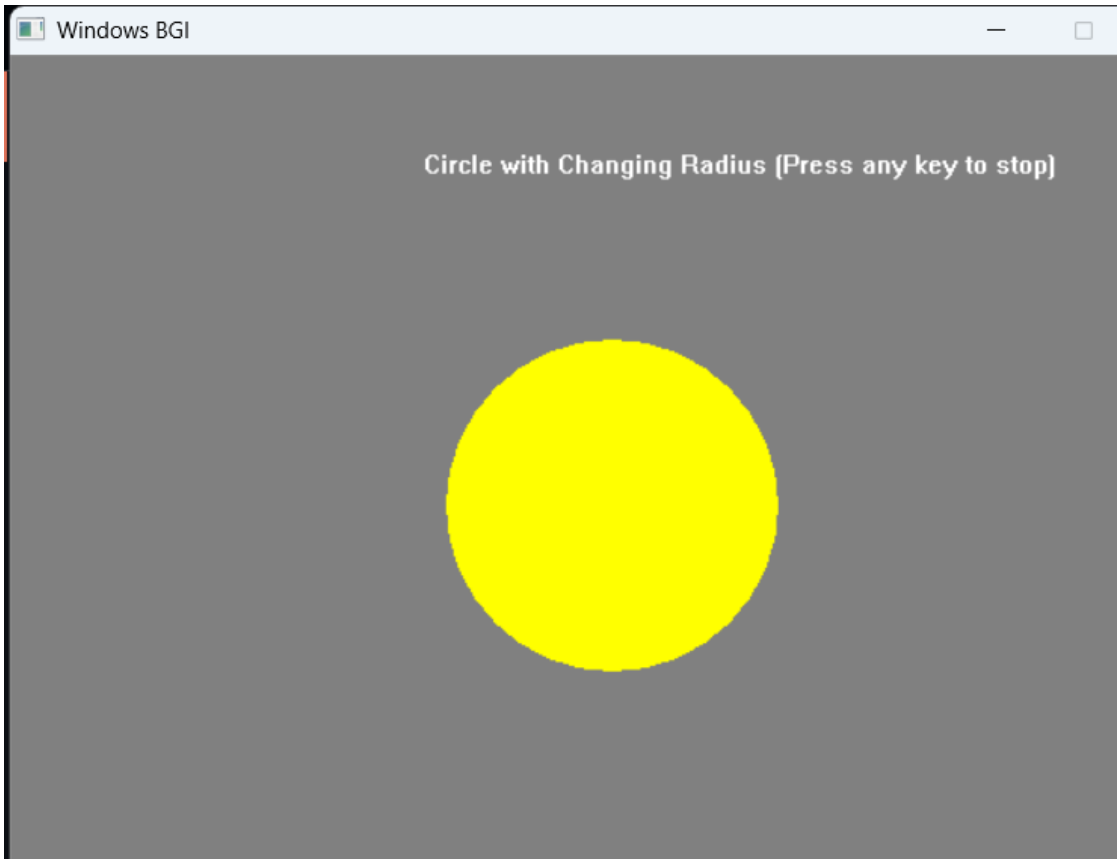
Practical: 19

Write a program to show changing radius of circle.

```
#include <graphics.h>
#include <conio.h>
#include <iostream>
using namespace std;

int main() {
   int gd = DETECT, gm;
    char data[] = "C:\\MinGW\\lib\\libbgi.a";
   initgraph(&gd, &gm, data);

   int midx = getmaxx() / 2;   // center X
   int midy = getmaxy() / 2;   // center Y
   int radius = 10;           // initial radius
   int step = 2;              // how much radius changes each frame
   int maxRadius = 150;       // maximum radius
   int minRadius = 10;        // minimum radius

   setbkcolor(DARKGRAY);
   cleardevice();

   outtextxy(midx - 100, 50, "Circle with Changing Radius (Press any key to stop)");

   // Animation loop
   while (!kbhit()) {
      cleardevice();
      setcolor(WHITE);
      outtextxy(midx - 100, 50, "Circle with Changing Radius (Press any key to stop)");

      // Draw circle with current radius
      setcolor(YELLOW);
      circle(midx, midy, radius);

      // Fill color
      setfillstyle(SOLID_FILL, YELLOW);
      floodfill(midx, midy, YELLOW);

      // Update radius
      radius += step;

      // Reverse direction when limits are reached
      if (radius >= maxRadius || radius <= minRadius)
```

```
        step = -step;

      delay(30);  // controls animation speed
   }

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

OUTPUT



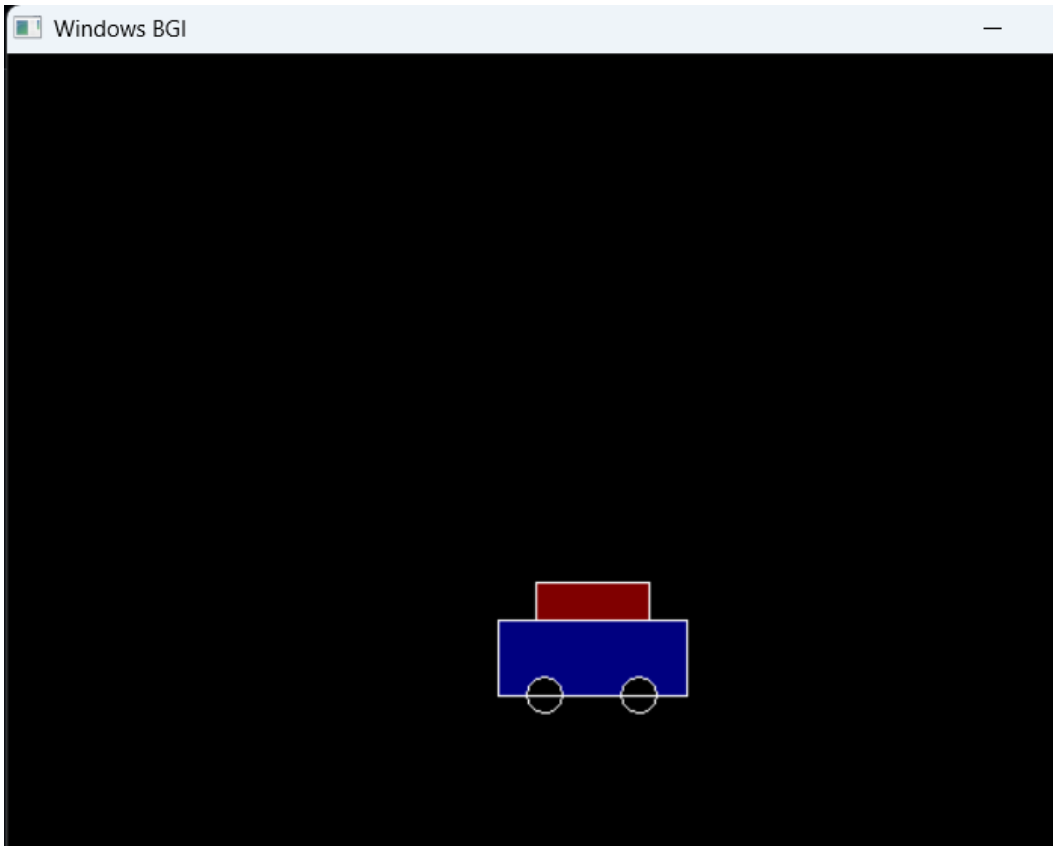
Practical: 20

Write a program to Display Moving Car.

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

void drawCar(int x, int y)
{
   // Car body
   setcolor(WHITE);
   rectangle(x, y, x + 100, y + 40);       // main body
   rectangle(x + 20, y - 20, x + 80, y);   // top

   // Wheels
   circle(x + 25, y + 40, 10);
   circle(x + 75, y + 40, 10);

   // Fill colors
   setfillstyle(SOLID_FILL, BLUE);
   floodfill(x + 2, y + 2, WHITE);

   setfillstyle(SOLID_FILL, RED);
   floodfill(x + 25, y - 5, WHITE);

   setfillstyle(SOLID_FILL, YELLOW);
   floodfill(x + 25, y + 40, WHITE);
   floodfill(x + 75, y + 40, WHITE);
}

int main()
{
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char*)"");

   int x = 0, y = 300;

   // Animate car movement
   for (x = 0; x <= 500; x += 5)
   {
      cleardevice();
      drawCar(x, y);
      delay(50);
   }
```

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

OUTPUT



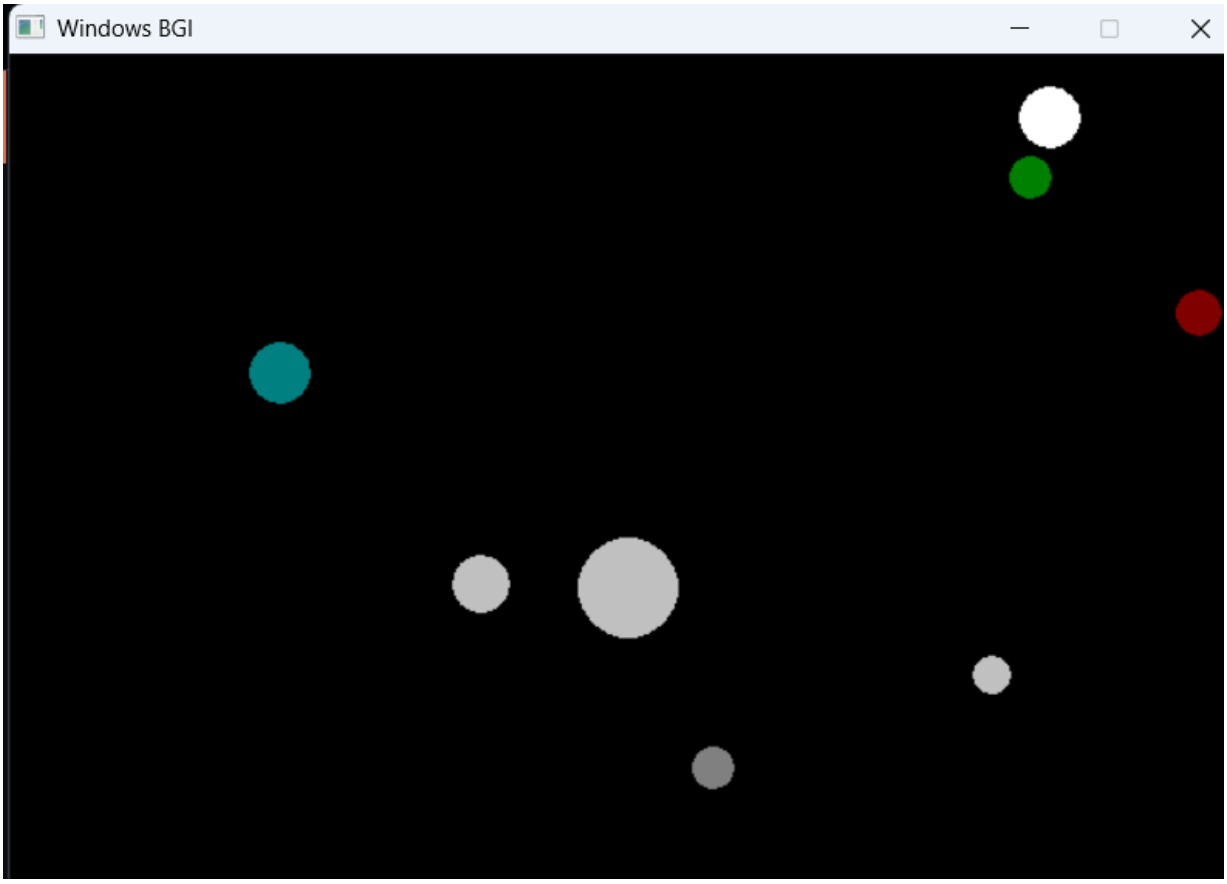
Practical: 21

Write a program for designing Screensaver.

```
#include <graphics.h>
#include <conio.h>
#include <stdlib.h>  // for rand()
#include <time.h>    // for seeding random numbers

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char*)"");

   // Get screen width and height
   int width = getmaxx();
   int height = getmaxy();

   // Seed random generator
   srand(time(0));

   const int balls = 10;
   int x[balls], y[balls], dx[balls], dy[balls], color[balls], radius[balls];

   // Initialize random positions and movement
   for (int i = 0; i < balls; i++) {
      x[i] = rand() % width;
      y[i] = rand() % height;
      dx[i] = (rand() % 6) + 2;
      dy[i] = (rand() % 6) + 2;
      color[i] = (rand() % 15) + 1;
      radius[i] = (rand() % 20) + 10;
   }

   // Animation loop
   while (!kbhit()) {
      cleardevice();

      for (int i = 0; i < balls; i++) {
         setcolor(color[i]);
         setfillstyle(SOLID_FILL, color[i]);
         circle(x[i], y[i], radius[i]);
         floodfill(x[i], y[i], color[i]);

         // Move the ball
         x[i] += dx[i];
```

```
        y[i] += dy[i];

        // Bounce off walls
        if (x[i] > width - radius[i] || x[i] < radius[i])
            dx[i] = -dx[i];
        if (y[i] > height - radius[i] || y[i] < radius[i])
            dy[i] = -dy[i];
      }

      delay(30); // smooth animation speed
   }

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

OUTPUT

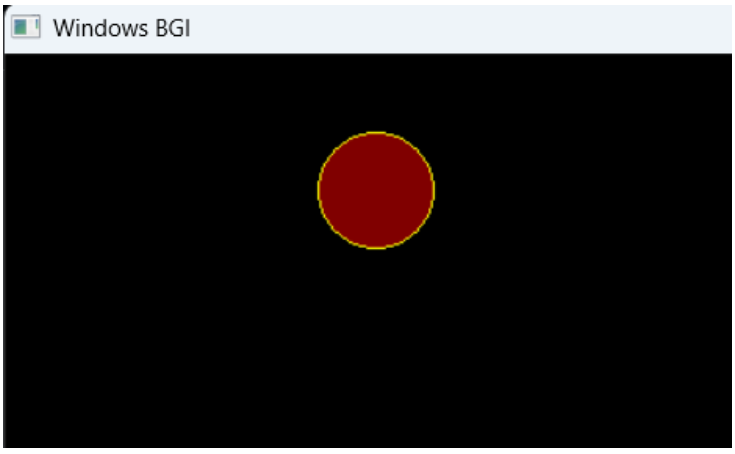


Practical: 22

Write a program to show Bouncing Ball Animation.

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

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char*)"");

   int x = 100, y = 100;      // initial position
   int radius = 30;           // ball radius
   int dx = 5, dy = 5;        // movement speed
   int maxX = getmaxx();      // screen width
   int maxY = getmaxy();      // screen height

   while (!kbhit()) {
      cleardevice();         // clear the previous frame
      setcolor(YELLOW);
      setfillstyle(SOLID_FILL, RED);
      circle(x, y, radius);
      floodfill(x, y, YELLOW);
      x += dx;
      y += dy;
      if (x + radius >= maxX || x - radius <= 0)
         dx = -dx;
      if (y + radius >= maxY || y - radius <= 0)
         dy = -dy;

      delay(20);     }
   getch();
   closegraph();
   return 0; }
```

OUTPUT



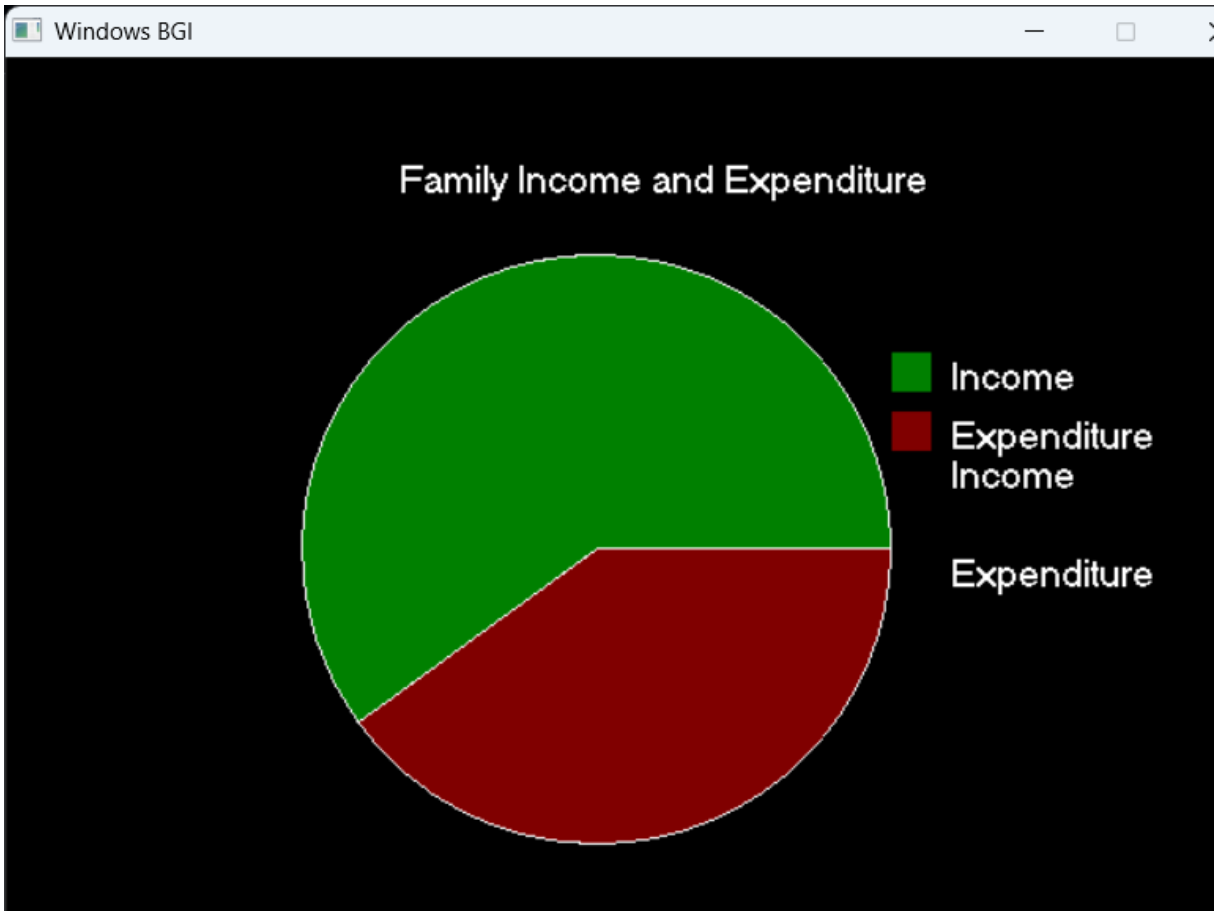
Practical: 23

Write a program to draw pie chart of Family income and Expenditure.

```
#include <graphics.h>
#include <conio.h>
#include <iostream>
using namespace std;

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char*)"");

   // Input income and expenditure data
   int income, expenditure;
   cout << "Enter Family Income: ";
   cin >> income;
   cout << "Enter Family Expenditure: ";
   cin >> expenditure;

   // Calculate total
   int total = income + expenditure;

   // Calculate percentage angles
   float income_angle = (income * 360.0) / total;
   float expenditure_angle = (expenditure * 360.0) / total;

   int x = 300, y = 250, radius = 150;
   settextstyle(3, 0, 2);

   // Draw title
   outtextxy(200, 50, (char*)"Family Income and Expenditure");

   // Draw Income sector
   setfillstyle(SOLID_FILL, GREEN);
   sector(x, y, 0, income_angle, radius, radius);
   outtextxy(x + 180, y - 50, (char*)"Income");

   // Draw Expenditure sector
   setfillstyle(SOLID_FILL, RED);
   sector(x, y, income_angle, income_angle + expenditure_angle, radius, radius);
   outtextxy(x + 180, y, (char*)"Expenditure");

   // Draw legend boxes
   setfillstyle(SOLID_FILL, GREEN);
```

```
    bar(450, 150, 470, 170);
    outtextxy(480, 150, (char*)"Income");

    setfillstyle(SOLID_FILL, RED);
    bar(450, 180, 470, 200);
    outtextxy(480, 180, (char*)"Expenditure");

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

OUTPUT

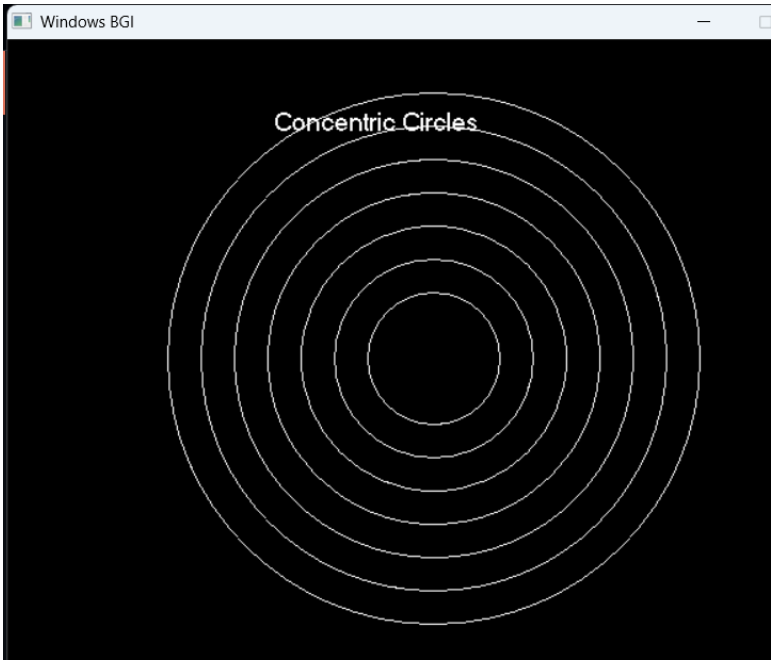


Practical: 24

Write a program to draw concentric circles on screen.

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

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char*)"");

   int x = 320, y = 240; // Center of screen
   int i;

   settextstyle(3, 0, 2);
   outtextxy(200, 50, (char*)"Concentric Circles");

   // Draw multiple concentric circles
   for (i = 50; i <= 200; i += 25) {
      circle(x, y, i);
      delay(300); // small delay for visual effect
   }

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

OUTPUT



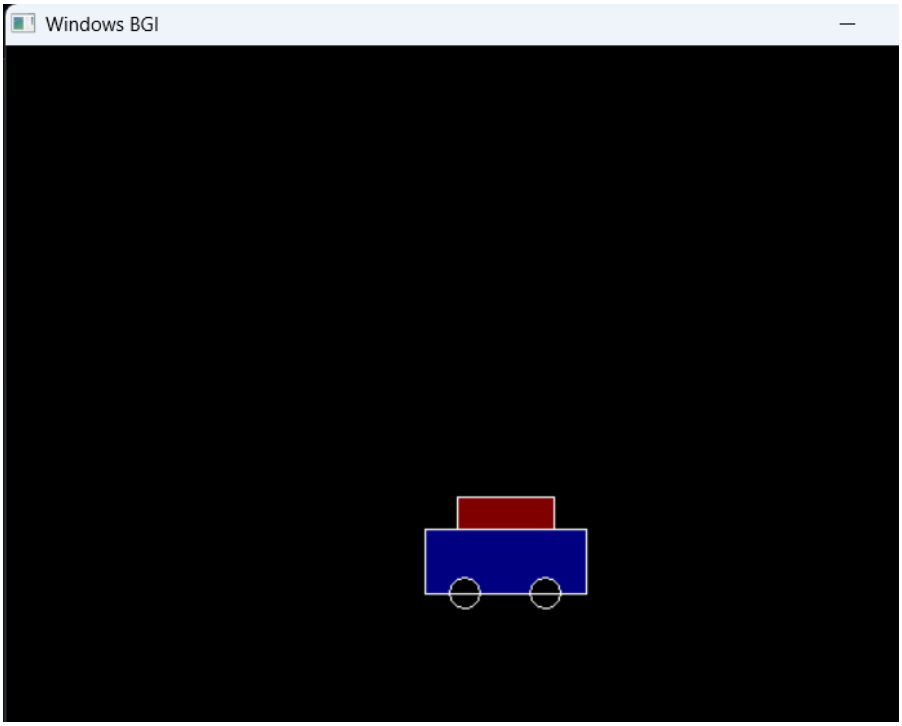
Practical: 25

Write a program to show moving car animation.

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

void drawCar(int x, int y)
{
   // Car body
   setcolor(WHITE);
   rectangle(x, y, x + 100, y + 40);       // main body
   rectangle(x + 20, y - 20, x + 80, y);   // top

   // Wheels
   circle(x + 25, y + 40, 10);
   circle(x + 75, y + 40, 10);

   // Fill colors
   setfillstyle(SOLID_FILL, BLUE);
   floodfill(x + 2, y + 2, WHITE);

   setfillstyle(SOLID_FILL, RED);
   floodfill(x + 25, y - 5, WHITE);

   setfillstyle(SOLID_FILL, YELLOW);
   floodfill(x + 25, y + 40, WHITE);
   floodfill(x + 75, y + 40, WHITE);
}

int main()
{
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char*)"");

   int x = 0, y = 300;

   // Animate car movement
   for (x = 0; x <= 500; x += 5)
   {
      cleardevice();
      drawCar(x, y);
      delay(50);
   }
```

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

OUTPUT



Practical: 26

Write a program to draw stars in night sky.

```
#include <graphics.h>
#include <conio.h>
#include <stdlib.h>   // for rand()
#include <time.h>     // for seeding rand()

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char*)"");

   // Set background color to dark blue (night sky)
   setbkcolor(BLUE);
   cleardevice();

   setcolor(WHITE);
   settextstyle(3, 0, 2);
   outtextxy(200, 30, (char*)"Night Sky with Twinkling Stars");

   srand(time(0));  // Seed random number generator

   // Draw static stars initially
   for (int i = 0; i < 100; i++) {
      int x = rand() % getmaxx();
      int y = rand() % getmaxy();
      putpixel(x, y, WHITE);
   }

   // Animate twinkling stars
   for (int t = 0; t < 300; t++) {
      int x = rand() % getmaxx();
      int y = rand() % getmaxy();
      int color = rand() % 15;  // random bright colors for twinkle

      putpixel(x, y, color);
      delay(50);
      putpixel(x, y, WHITE); // revert to white after twinkle
   }

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

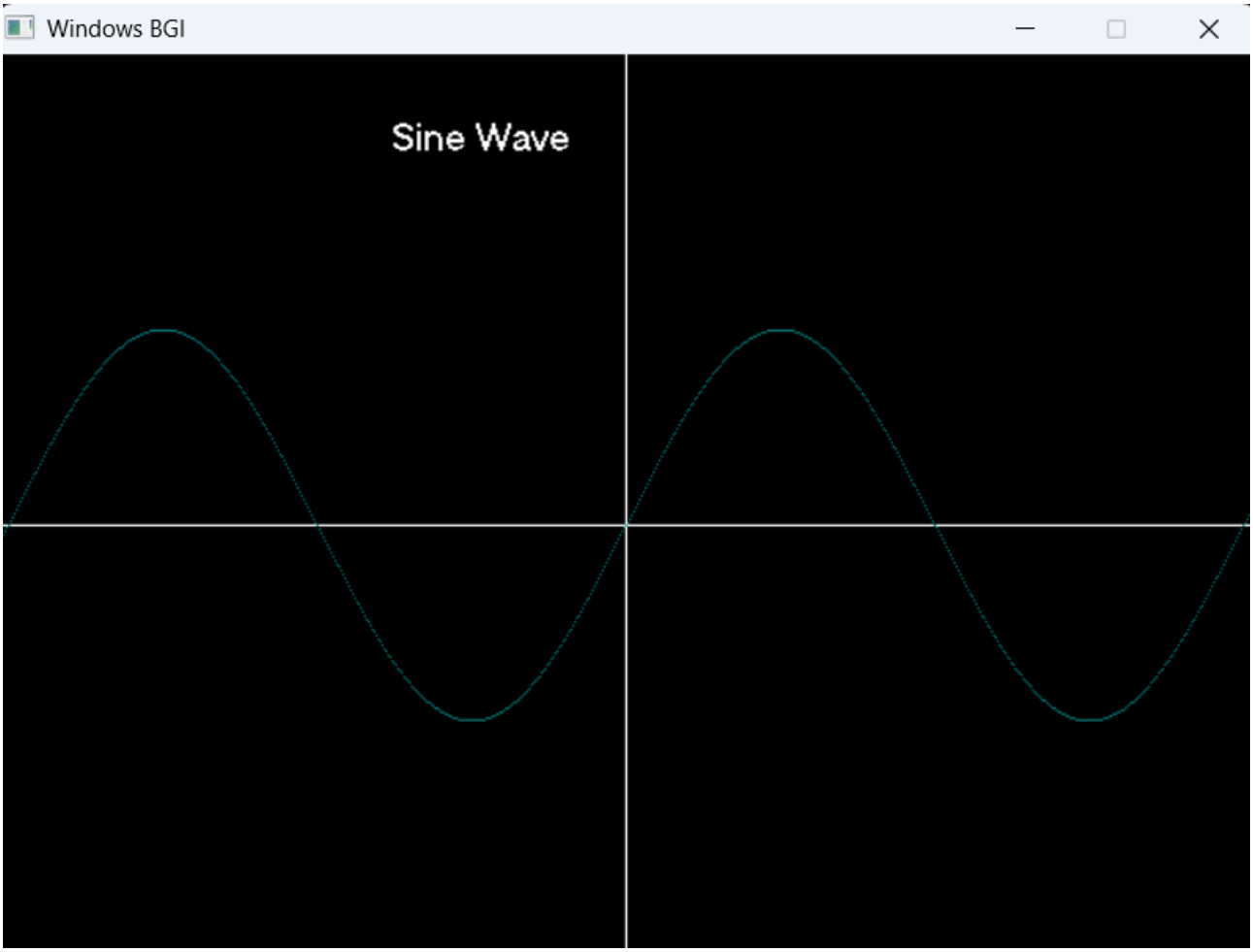
OUTPUT

Practical: 27

Write a program to draw Sine Wave.

```
#include <graphics.h>
#include <conio.h>
#include <math.h>   // for sin() and M_PI

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char*)"");

   setbkcolor(BLACK);
   cleardevice();

   setcolor(WHITE);
   settextstyle(3, 0, 2);
   outtextxy(200, 30, (char*)"Sine Wave");

   // Draw X and Y axes
   int midx = getmaxx() / 2;
   int midy = getmaxy() / 2;

   line(0, midy, getmaxx(), midy);    // X-axis
   line(midx, 0, midx, getmaxy());    // Y-axis

   setcolor(YELLOW);

   // Draw the sine wave
   for (int x = 0; x < getmaxx(); x++) {
      // Convert pixel x to radians
      double angle = (x - midx) * 0.02;   // Adjust 0.02 for frequency
      int y = midy - (int)(100 * sin(angle));  // Amplitude = 100 pixels
      putpixel(x, y, RED);
      delay(5); // small delay to visualize drawing
   }

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

OUTPUT

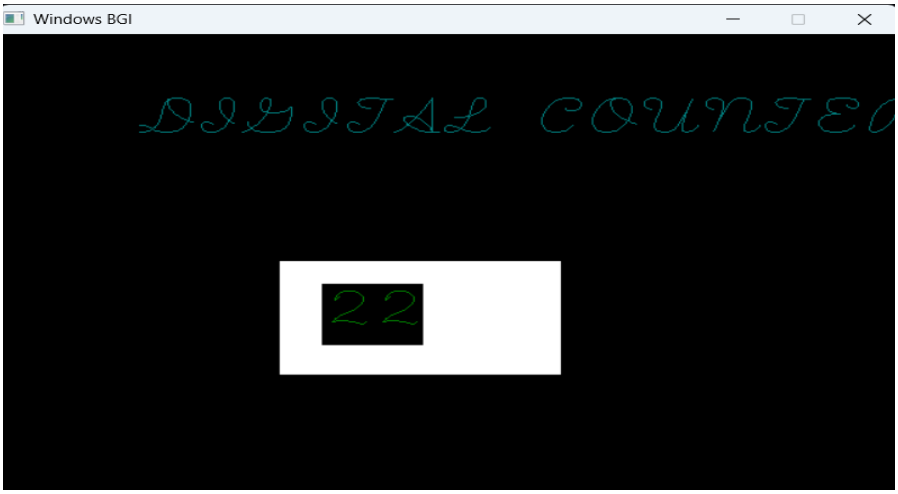


Practical: 28

Write a program to make a Digital counter.

```
#include <graphics.h>
#include <conio.h>
#include <dos.h>  // for delay()
int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char*)"");

   setbkcolor(BLACK);
   cleardevice();
   setcolor(CYAN);
   settextstyle(5, 0, 6); // 7 = digital font style

   outtextxy(100, 50, (char*)"DIGITAL COUNTER");

   for (int i = 0; i <= 99; i++) {
      char num[10];
      sprintf(num, "%02d", i);  // format number as 2 digits (e.g., 01, 09, 10)

      setcolor(BLACK);
      bar(200, 200, 400, 300)
      setcolor(GREEN);
      outtextxy(230, 220, num);
      delay(500); // wait 0.5 seconds
   }
   getch();
   closegraph();
   return 0;
}
```

OUTPUT



Practical: 29

Write a program to show Flag Hoisting.

```
#include <graphics.h>
#include <conio.h>
#include <dos.h>  // for delay()

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char*)"");
   setbkcolor(LIGHTBLUE);
   cleardevice();

   int baseX = 200, baseY = 400; // Flag pole base
   int poleHeight = 300;
   int flagWidth = 180, flagHeight = 90;
   int y; // current flag top Y position

   setcolor(DARKGRAY);
   rectangle(baseX - 20, baseY, baseX + 20, baseY + 20);
   setfillstyle(SOLID_FILL, DARKGRAY);
   floodfill(baseX, baseY + 10, DARKGRAY);

   // Draw flag pole
   setcolor(WHITE);
   line(baseX, baseY, baseX, baseY - poleHeight);

   // Animate flag hoisting (moving upward)
   for (y = baseY - 10; y >= baseY - poleHeight + flagHeight; y -= 5) {
      // Clear previous flag area
      setfillstyle(SOLID_FILL, LIGHTBLUE);
      bar(baseX + 1, baseY - poleHeight, baseX + flagWidth + 2, baseY);

      // Draw saffron (top)
      setcolor(RED);
      rectangle(baseX, y, baseX + flagWidth, y - flagHeight / 3);
      setfillstyle(SOLID_FILL, RED);
      floodfill(baseX + 5, y - 5, RED);

      // Draw white (middle)
      setcolor(WHITE);
      rectangle(baseX, y - flagHeight / 3, baseX + flagWidth, y - (2 * flagHeight / 3));
      setfillstyle(SOLID_FILL, WHITE);
      floodfill(baseX + 5, y - flagHeight / 2, WHITE);
```

```
        setcolor(GREEN);
        rectangle(baseX, y - (2 * flagHeight / 3), baseX + flagWidth, y - flagHeight);
        setfillstyle(SOLID_FILL, GREEN);
        floodfill(baseX + 5, y - (2 * flagHeight / 3) - 5, GREEN);

        // Draw blue Ashoka Chakra in center of white part
        setcolor(BLUE);
        circle(baseX + flagWidth / 2, y - flagHeight / 2, 10);
        for (int i = 0; i < 24; i++) {
            float angle = i * (360 / 24.0) * 3.14 / 180;
            line(baseX + flagWidth / 2, y - flagHeight / 2,
                 baseX + flagWidth / 2 + 10 * cos(angle),
                 y - flagHeight / 2 + 10 * sin(angle));
        }

        delay(100); // slow upward motion
    }
    settextstyle(3, 0, 3);
    setcolor(BLACK);
    outtextxy(100, 440, (char*)"Happy Independence Day!");

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

OUTPUT

Practical: 30

Write a program to Draw a moving cycle.

```
#include <graphics.h>
#include <conio.h>
#include <math.h>
#include <direct.h>  // for delay()

int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, (char*)"");

   setbkcolor(LIGHTBLUE);
   cleardevice();

   int x, y = 350; // ground position (y-coordinate for wheels)

   // Ground line
   setcolor(GREEN);
   line(0, y + 50, getmaxx(), y + 50);

   // Move the cycle from left to right
   for (x = 0; x < getmaxx() - 200; x += 5) {
      // Clear previous frame
      cleardevice();

      // Redraw ground
      setcolor(GREEN);
      line(0, y + 50, getmaxx(), y + 50);

      // Cycle body color
      setcolor(RED);

      // Two wheels
      circle(x + 50, y, 30);   // back wheel
      circle(x + 150, y, 30);  // front wheel

      // Frame (triangle body)
      line(x + 50, y, x + 100, y - 40);
      line(x + 100, y - 40, x + 150, y);
      line(x + 50, y, x + 150, y);

      // Handle
      line(x + 150, y, x + 170, y - 40);
```

```
        line(x + 170, y - 40, x + 160, y - 45);

        // Seat
        line(x + 80, y - 40, x + 110, y - 40);
        line(x + 90, y - 40, x + 90, y - 50);

        // Pedal (rotating effect)
        static int angle = 0;
        int pedalX1 = x + 100 + 10 * cos(angle * 3.14 / 180);
        int pedalY1 = y + 10 * sin(angle * 3.14 / 180);
        int pedalX2 = x + 100 - 10 * cos(angle * 3.14 / 180);
        int pedalY2 = y - 10 * sin(angle * 3.14 / 180);
        line(pedalX1, pedalY1, pedalX2, pedalY2);
        circle(pedalX1, pedalY1, 2);
        circle(pedalX2, pedalY2, 2);

        angle = (angle + 10) % 360;

        delay(50);
    }

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

OUTPUT