VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES
VIVEKANANDA SCHOOL OF INFORMATION TECHNOLOGY

BACHELOR OF COMPUTER APPLICATION
Practical- CG LAB File
BCA-373

Guru Gobind Singh Indraprastha University
Sector - 16C Dwarka, Delhi – 110078

SUBMITTED TO:
Dr. Neha Goel
Associate Professor
VSIT

SUBMITTED BY:
Tushar Arora
00529802021
BCA 5EA

INDEX

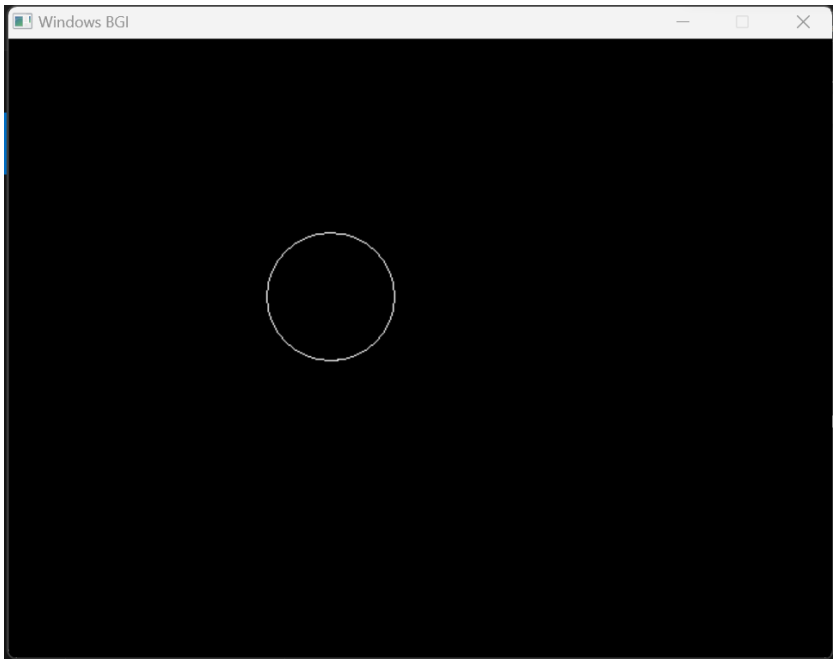
S NO.	Practical's	Signature
1.	Drawing objects like circle, rectangle, polygon using graphic function	
2.	Line Drawing Algorithms (DDA & Bresenham's Algorithm)	
3.	Circle Algorithms	
4.	Translation in 2D	
5.	Rotation in 2D	
6.	Scaling in 2D	
7.	Shearing in 2D	
8.	Cohen Sutherland's Algorithm	
9.	Graphics Inbuilt functions	
10.	Reflection in 2D	
11.	Program to rotate a circle outside another circle	
12.	Program to draw Flying Balloons	
13.	Show Bouncing Ball Animation	
14.	Draw a moving cycle	
15.	Show Moving Car Animation	

Ques. Drawing objects like circle, rectangle, polygon using graphic function

Code –

```
#include <graphics.h>
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");
    circle(250, 200, 50);
    getch();
    closegraph();
    return 0;
}
```

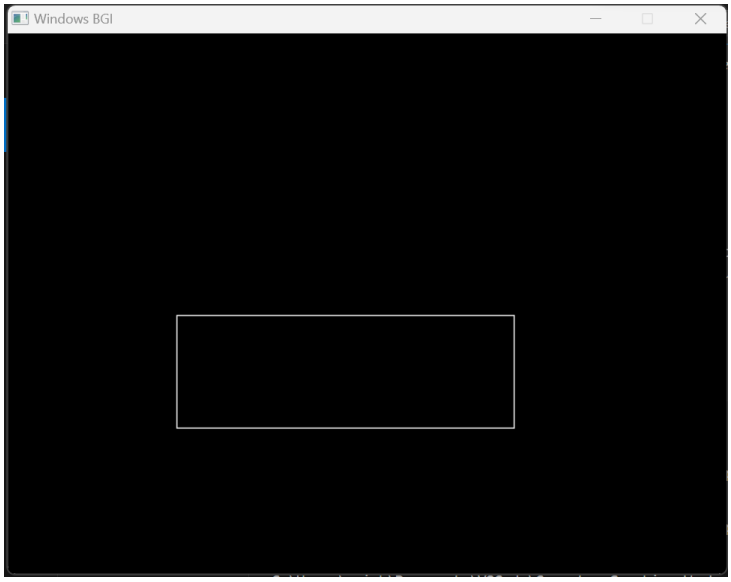
Output –



Code –

```
#include <graphics.h>
int main() {
   int gd = DETECT, gm;
   int left = 150, top = 250;
   int right = 450, bottom = 350;
   initgraph(&gd, &gm, "");
   rectangle(left, top, right, bottom);
   getch();
   closegraph();
   return 0;
}
```

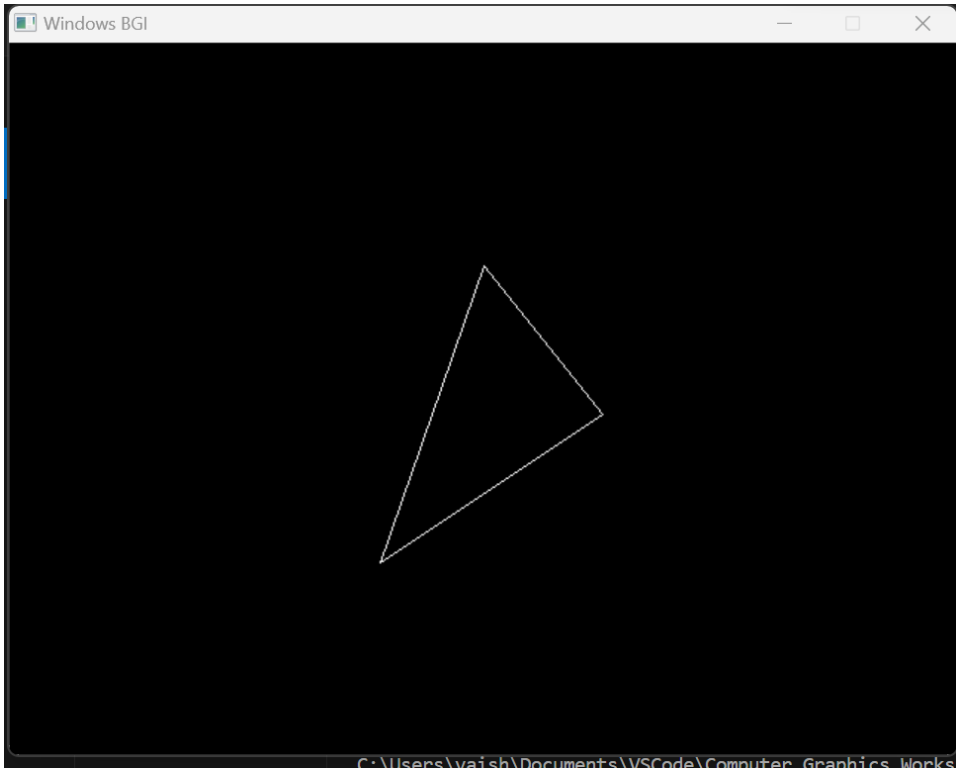
Output –



Code –

```
#include <graphics.h>
int main() {
    int gd = DETECT, gm;
    int arr[] = {320, 150, 400, 250, 250, 350, 320, 150};
    initgraph(&gd, &gm, "");
    drawpoly(4, arr);
    getch();
    closegraph();
    return 0;
}
```

Output –

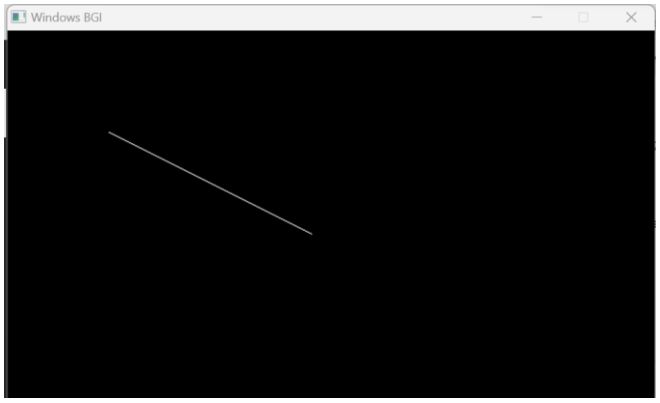


Ques. Line Drawing Algorithms (DDA & Bresenham's Algorithm)

Code – DDA

```
#include <graphics.h>
void drawLineDDA(int X1, int Y1, int X2, int Y2) {
   int dx = X2 - X1;
   int dy = Y2 - Y1;
   int steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);
   float Xinc = dx / (float) steps;
   float Yinc = dy / (float) steps;
   float X = X1;
   float Y = Y1;
   for (int i = 0; i <= steps; i++) {
      putpixel(X, Y, WHITE);  // assuming WHITE is defined as 15 in graphics.h
      X += Xinc;
      Y += Yinc;
   }
}
int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");
   int X1 = 100, Y1 = 100, X2 = 300, Y2 = 200;
   drawLineDDA(X1, Y1, X2, Y2);
   getch();
   closegraph();
   return 0;
}
```

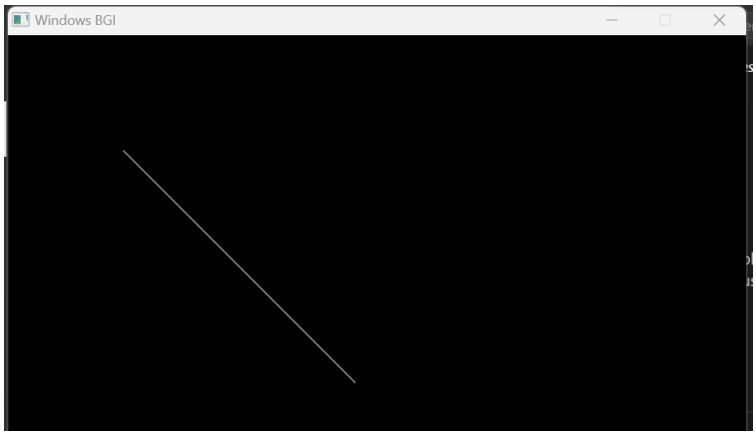
Output –



Code – Bresenham's

```
#include <graphics.h>
void drawLineBresenham(int X1, int Y1, int X2, int Y2) {
   int dx = abs(X2 - X1), dy = abs(Y2 - Y1);
   int slope_less_than_one = dy <= dx;
   int interchange_needed = 0;
   if (!slope_less_than_one && (interchange_needed = 1))
      dx += dy, dy = dx - dy, dx -= dy;
   int p = 2 * dy - dx;
   int twoDy = 2 * dy, twoDyMinusDx = 2 * (dy - dx);
   for (int X = X1, Y = Y1; X <= X2; X++, p += slope_less_than_one ? twoDy :
twoDyMinusDx) {
      putpixel(slope_less_than_one ? X : Y, slope_less_than_one ? Y : X, WHITE);
      if (p >= 0)
         Y++, p -= slope_less_than_one ? dx : twoDyMinusDx;
   }
}
int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");
   int X1 = 100, Y1 = 100, X2 = 300, Y2 = 200;
   drawLineBresenham(X1, Y1, X2, Y2);
   getch();
   closegraph();
   return 0;
}
```

Output –

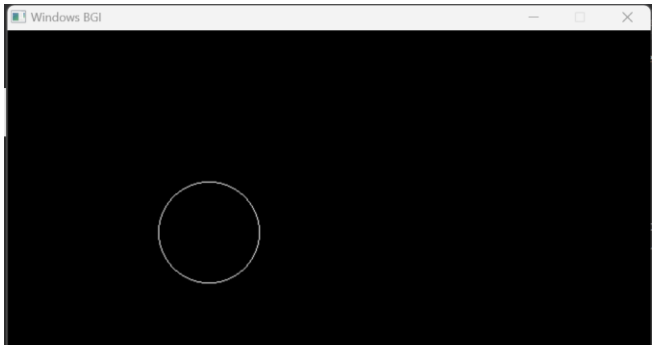


Ques. Circle Algorithms

Code – Bresenham

```
#include <graphics.h>
void drawCircleBresenham(int xc, int yc, int r) {
    int x = 0, y = r;
    int p = 3 - 2 * r;
    while (x <= y) {
        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);
        if (p < 0)
            p += 4 * x++ + 6;
        else
            p += 4 * (x++ - y--) + 10;
    }
}
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");
    int xc = 200, yc = 200, r = 50;
    drawCircleBresenham(xc, yc, r);
    getch();
    closegraph();
    return 0;
}
```

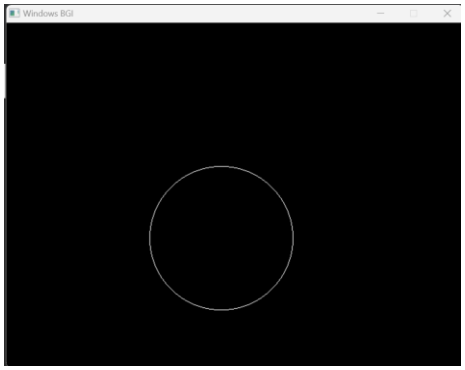
Output –



Code – Midpoint Circle

```
 #include <graphics.h>
void drawCircleMidpoint(int xc, int yc, int r) {
    int x = r, y = 0;
    int p = 1 - r;
    while (x >= y) {
        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);
        y++;
        if (p <= 0)
            p = p + 2*y + 1;
        else {
            x--;
            p = p + 2*y - 2*x + 1;
      }
    }
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");
    int xc = 200, yc = 200, r = 50;
    drawCircleMidpoint(xc, yc, r);
    getch();
    closegraph();
    return 0;
}
```

Output –

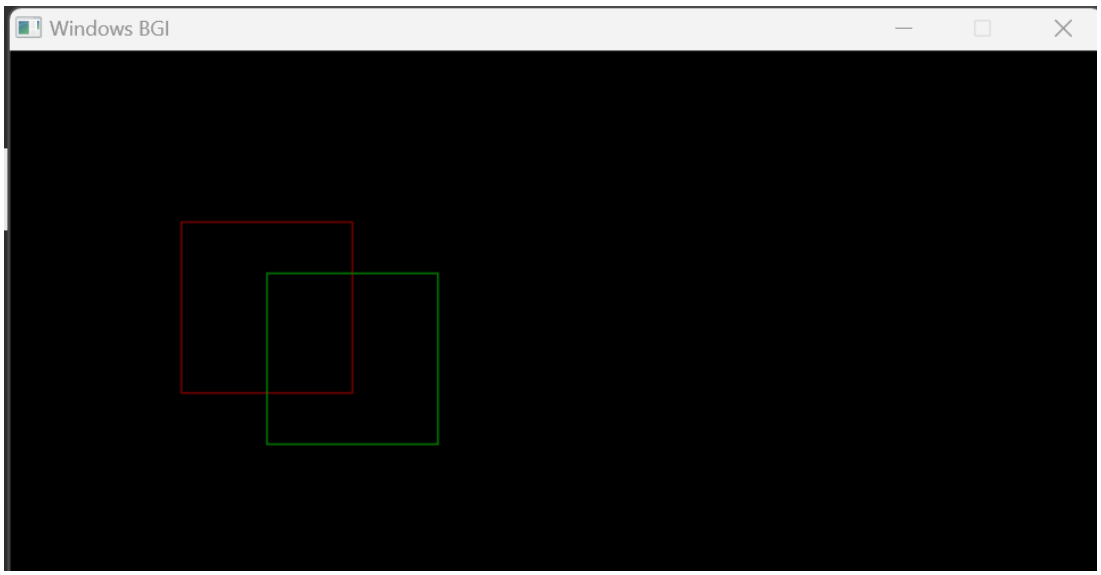


Ques. Translation in 2D

Code –

```
#include <graphics.h>
void drawSquare(int x, int y, int side) {
   rectangle(x, y, x + side, y + side);
}
void translateSquare(int x, int y, int tx, int ty, int side) {
   setcolor(RED);
   drawSquare(x, y, side);
   setcolor(GREEN);
   drawSquare(x + tx, y + ty, side);
}
int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");
   int x = 100, y = 100, side = 50;
   int tx = 50, ty = 30;
   translateSquare(x, y, tx, ty, side);
   getch();
   closegraph();
   return 0;
}
```

Output –

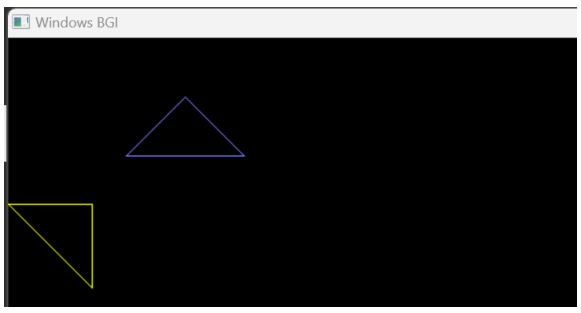


Ques. Rotation in 2D

Code –

```
#include <graphics.h>
#include <math.h>
void drawTriangle(int x1, int y1, int x2, int y2, int x3, int y3, int color) {
   setcolor(color);
   line(x1, y1, x2, y2);
   line(x2, y2, x3, y3);
   line(x3, y3, x1, y1);
}
void rotateTriangle(int x1, int y1, int x2, int y2, int x3, int y3, int angle) {
   drawTriangle(x1, y1, x2, y2, x3, y3, LIGHTBLUE);
   setcolor(YELLOW);
   float angleRad = angle * M_PI / 180;
   int new_x1 = round(x1 * cos(angleRad) - y1 * sin(angleRad));
   int new_y1 = round(x1 * sin(angleRad) + y1 * cos(angleRad));
   int new_x2 = round(x2 * cos(angleRad) - y2 * sin(angleRad));
   int new_y2 = round(x2 * sin(angleRad) + y2 * cos(angleRad));
   int new_x3 = round(x3 * cos(angleRad) - y3 * sin(angleRad));
   int new_y3 = round(x3 * sin(angleRad) + y3 * cos(angleRad));
   drawTriangle(new_x1, new_y1, new_x2, new_y2, new_x3, new_y3, YELLOW);  }
int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");
   int x1 = 100, y1 = 100;
   int x2 = 200, y2 = 100;
   int x3 = 150, y3 = 50;
   int angle = 45;
   rotateTriangle(x1, y1, x2, y2, x3, y3, angle);
   getch();
   closegraph();
   return 0;
}
```

Output –

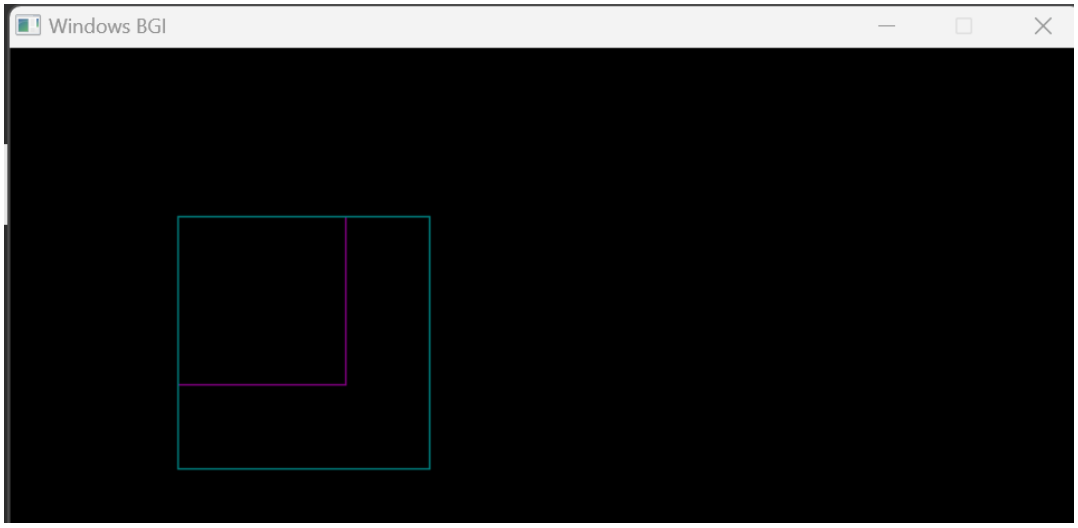


Ques. Scaling in 2D

Code –

```
#include <graphics.h>
void drawSquare(int x, int y, int side) {
    rectangle(x, y, x + side, y + side);
}
void scaleSquare(int x, int y, float scale_factor, int side) {
    setcolor(MAGENTA);
    drawSquare(x, y, side);  // Original square in magenta
    setcolor(CYAN);
    drawSquare(x, y, scale_factor * side);  // Scaled square in cyan
}
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");
    int x = 100, y = 100, side = 100;  // Larger square
    float scale_factor = 1.5;
    scaleSquare(x, y, scale_factor, side);
    getch();
    closegraph();
    return 0;
}
```

Output –

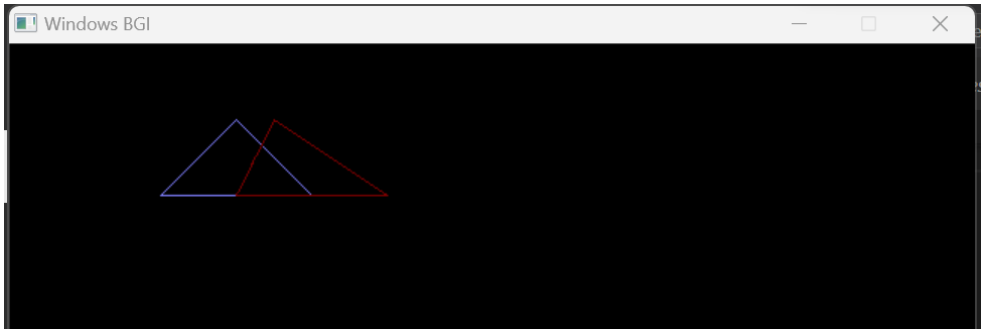


Ques. Shearing in 2D

Code –

```
#include <graphics.h>
#include <stdio.h>
void drawTriangle(int x1, int y1, int x2, int y2, int x3, int y3, int color) {
   setcolor(color);
   line(x1, y1, x2, y2);
   line(x2, y2, x3, y3);
   line(x3, y3, x1, y1);
}
void shearTriangle(int x1, int y1, int x2, int y2, int x3, int y3, float shearFactor) {
   drawTriangle(x1, y1, x2, y2, x3, y3, LIGHTBLUE);
   setcolor(RED);
   int new_x1 = x1 + shearFactor * y1;
   int new_y1 = y1;
   int new_x2 = x2 + shearFactor * y2;
   int new_y2 = y2;
   int new_x3 = x3 + shearFactor * y3;
   int new_y3 = y3;
   drawTriangle(new_x1, new_y1, new_x2, new_y2, new_x3, new_y3, RED);
}
int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");
   int x1 = 100, y1 = 100;
   int x2 = 200, y2 = 100;
   int x3 = 150, y3 = 50;
   float shearFactor = 0.5;
   shearTriangle(x1, y1, x2, y2, x3, y3, shearFactor);
   getch();
   closegraph();
   return 0;
}
```

Output –



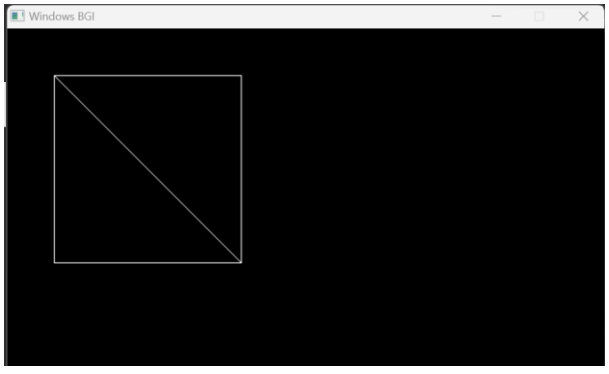
Ques. Cohen Sutherland's Algorithm

Code –

```
#include <graphics.h>
#define INSIDE 0 // 0000
#define LEFT 1   // 0001
#define RIGHT 2  // 0010
#define BOTTOM 4 // 0100
#define TOP 8    // 1000
int x_max, y_max, x_min, y_min;
int computeCode(int x, int y) {
   int code = INSIDE;
   if (x < x_min)
      code |= LEFT;
   else if (x > x_max)
      code |= RIGHT;
   if (y < y_min)
      code |= BOTTOM;
   else if (y > y_max)
      code |= TOP;
   return code;
}
void cohenSutherland(int x1, int y1, int x2, int y2) {
   int code1 = computeCode(x1, y1);
   int code2 = computeCode(x2, y2);
   int accept = 0;
   while (1) {
      if (!(code1 | code2)) {
         accept = 1;
         break;
      } else if (code1 & code2) {
         break;
      } else {
         int code_out;
         int x, y;
         if (code1 != 0)
            code_out = code1;
         else
            code_out = code2;
         if (code_out & TOP) {
            x = x1 + (x2 - x1) * (y_max - y1) / (y2 - y1);
            y = y_max;
         } else if (code_out & BOTTOM) {
```

```
            x = x1 + (x2 - x1) * (y_min - y1) / (y2 - y1);
            y = y_min;
         } else if (code_out & RIGHT) {
            y = y1 + (y2 - y1) * (x_max - x1) / (x2 - x1);
            x = x_max;
         } else if (code_out & LEFT) {
            y = y1 + (y2 - y1) * (x_min - x1) / (x2 - x1);
            x = x_min;
         }
         if (code_out == code1) {
            x1 = x;
            y1 = y;
            code1 = computeCode(x1, y1);
         } else {
            x2 = x;
            y2 = y;
            code2 = computeCode(x2, y2);
         }
      }
   }
int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");
   x_min = 50;
   y_min = 50;
   x_max = 250;
   y_max = 250;
   rectangle(x_min, y_min, x_max, y_max);
   cohenSutherland(30, 30, 300, 300);
   getch();
   closegraph();
   return 0;
}
```

Output –



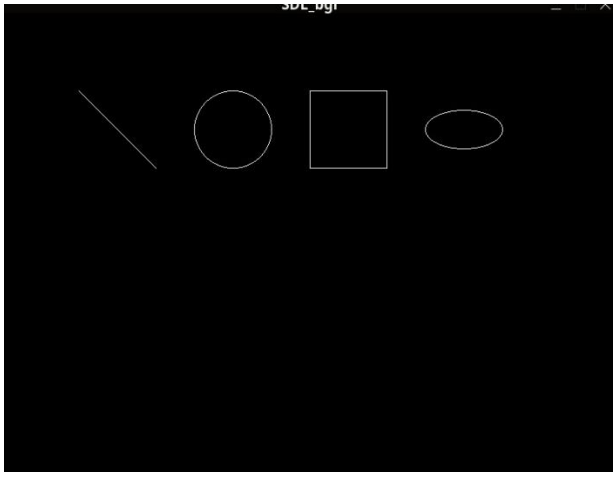
Ques. Graphics Inbuilt functions

Code –

```
#include <graphics.h>

int main()
{
   int gd = DETECT, gm;
   initgraph(&gd, &gm, NULL);
   line(100, 100, 200, 200);
   circle(300, 150, 50);
   rectangle(400, 100, 500, 200);
   ellipse(600, 150, 0, 360, 50, 25);
   getch();
   closegraph();
}
```

Output –



Ques. Reflection in 2D

Code –

```
#include <graphics.h>
void drawOriginalShape() {
   setcolor(RED);
   rectangle(100, 100, 200, 200);
}
void drawReflectedShape() {
   setcolor(BLUE);
   rectangle(100, getmaxy() - 100, 200, getmaxy() - 200);
}
int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, NULL);
   drawOriginalShape();
   delay(2000);
   cleardevice();
   drawReflectedShape();
   getch();
   closegraph();
   return 0;
}
```

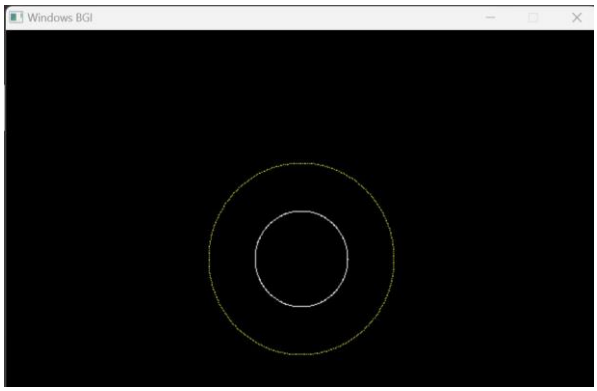
Output –

Ques. Program to rotate a circle outside another circle

Code –

```
#include <graphics.h>
#include <math.h>
void drawRotatingCircle(int xc, int yc, int outerRadius, int innerRadius, float outerAngle,
float innerAngle) {
   float theta;
   int x_outer, y_outer, x_inner, y_inner;
 for (theta = 0; theta <= 360; theta += 1) {
      x_outer = xc + outerRadius * cos((theta + outerAngle) * 3.14 / 180);
      y_outer = yc + outerRadius * sin((theta + outerAngle) * 3.14 / 180);
      putpixel(x_outer, y_outer, YELLOW);
   }          for (theta = 0; theta <= 360; theta += 1) {
      x_inner = xc + innerRadius * cos((theta + innerAngle) * 3.14 / 180);
      y_inner = yc + innerRadius * sin((theta + innerAngle) * 3.14 / 180);
      putpixel(x_inner, y_inner, WHITE);
   }}
int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");
   int xc = getmaxx() / 2;
   int yc = getmaxy() / 2;
   int outerRadius = 100; int innerRadius = 50;float outerAngle = 0;float innerAngle = 0;
   while (!kbhit()) {
      cleardevice();
      drawRotatingCircle(xc, yc, outerRadius, innerRadius, outerAngle, innerAngle);
      innerAngle -= 3;
   }
   getch();
   closegraph();
   return 0;
}
```

Output –

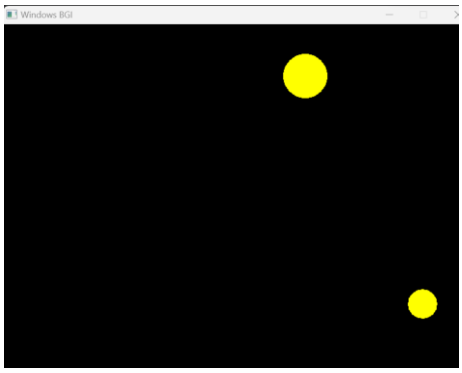


Ques. Program to draw Flying Balloons

Code –

```
#include <graphics.h>
#include <stdlib.h>
void drawBalloon(int x, int y, int radius) {
   setcolor(YELLOW);
   setfillstyle(SOLID_FILL, YELLOW);
   circle(x, y, radius);
   floodfill(x, y, YELLOW);
}
void moveBalloon(int &x, int &y, int speed) {
   x += speed;
   if (x > getmaxx() + 50) {
      x = -50;
      y = rand() % (getmaxy() - 50) + 50;
   }
}
int main() {
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "");
   int x1 = 50, y1 = 100;
   int x2 = 150, y2 = 200;
    for (int i = 0; i < 1000; ++i) {
      cleardevice();
      drawBalloon(x1, y1, 20);
      drawBalloon(x2, y2, 30);
      moveBalloon(x1, y1, 2);
      moveBalloon(x2, y2, 1);
      delay(10);
   }
   getch();
   closegraph();
   return 0;
}
```

Output –

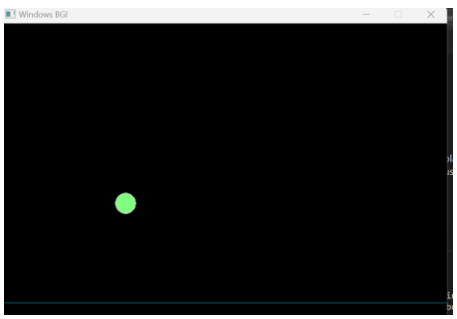


Ques. Show Bouncing Ball Animation

Code –

```
#include <graphics.h>
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);
    int x = 20, y = 200, dy = 2, uplimit = 250;
    while (true) {
        cleardevice();
        setcolor(3);
        line(0, 400, 679, 400);
        y += dy;
        x += 1;
        if (y >= 385 && dy > 0) {
            dy = -dy;
        } else if (y <= uplimit && dy < 0) {
            dy = -dy;
            uplimit += 20;
        }
        setcolor(15);
        fillellipse(x, y, 15, 15);
        delay(15);
        setcolor(0);
        setfillstyle(1, 10);
        fillellipse(x, y, 15, 15);
    }
    getch();
    closegraph();
    return 0;
}
```

Output –

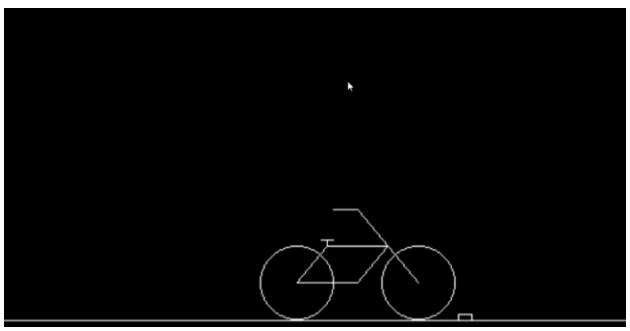


Ques. Draw a moving cycle

Code –

```
#include <conio.h>
#include <dos.h>
#include <graphics.h>
#include <iostream.h>
int main() {
        int gd = DETECT, gm, i, a;
        initgraph(&gd, &gm, );
        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);
                circle(150 + i, 405, 30);
                circle(50 + i, 405, 30);
                line(0, 436, getmaxx(), 436);
                rectangle(getmaxx() - i, 436,
                                650 - i, 431);
                delay(10);
                cleardevice();
        }
        getch();
        closegraph();
}
```

Output –



Ques. Show Moving Car Animation.

Code –

```
#include <graphics.h>
#include <stdio.h>
void draw_moving_car(void) {
   int i, j = 0, gd = DETECT, gm;
   initgraph(&gd, &gm, "");
   for (i = 0; i <= 420; i = i + 10) {
      setcolor(RED);
         line(0 + i, 300, 210 + i, 300);
         line(50 + i, 300, 75 + i, 270);
         line(75 + i, 270, 150 + i, 270);
         line(150 + i, 270, 165 + i, 300);
         line(0 + i, 300, 0 + i, 330);
         line(210 + i, 300, 210 + i, 330);
         circle(65 + i, 330, 15);
         circle(65 + i, 330, 2);
         circle(145 + i, 330, 15);
         circle(145 + i, 330, 2);
         line(0 + i, 330, 50 + i, 330);
         line(80 + i, 330, 130 + i, 330);
         line(210 + i, 330, 160 + i, 330);
         delay(100);
      setcolor(BLACK);
      line(210 + i, 300, 210 + i, 330);
      circle(65 + i, 330, 15);
      circle(65 + i, 330, 2);
      circle(145 + i, 330, 15);
      circle(145 + i, 330, 2);
      line(0 + i, 330, 50 + i, 330);
      line(80 + i, 330, 130 + i, 330);
      line(210 + i, 330, 160 + i, 330);}
   getch();
   closegraph();}
int main() {
   draw_moving_car();
   return 0;}
```

Output –