Course Title: *Computer Graphics Laboratory*

Course code: *CSE-304*

3rd year 1st semester

Date of Submission: 04/06/2023

Submitted to-

Dr. Mohammad Shorif Uddin

Professor

and

Dr. Morium Akter

Associate Professor
Department of Computer Science and Engineering

Jahangirnagar University

Savar, Dhaka-1342

Sl	Class Roll	Exam Roll	Name
01	395	202207	Md. Mahfuz Molla

Department of Computer Science and Engineering
Jahangirnagar University
Savar, Dhaka, Bangladesh

Experiment No : 01

Name of Experiment: *Scan convert a Line From p1(0,0) To p2(100,50) and perform specific task…*

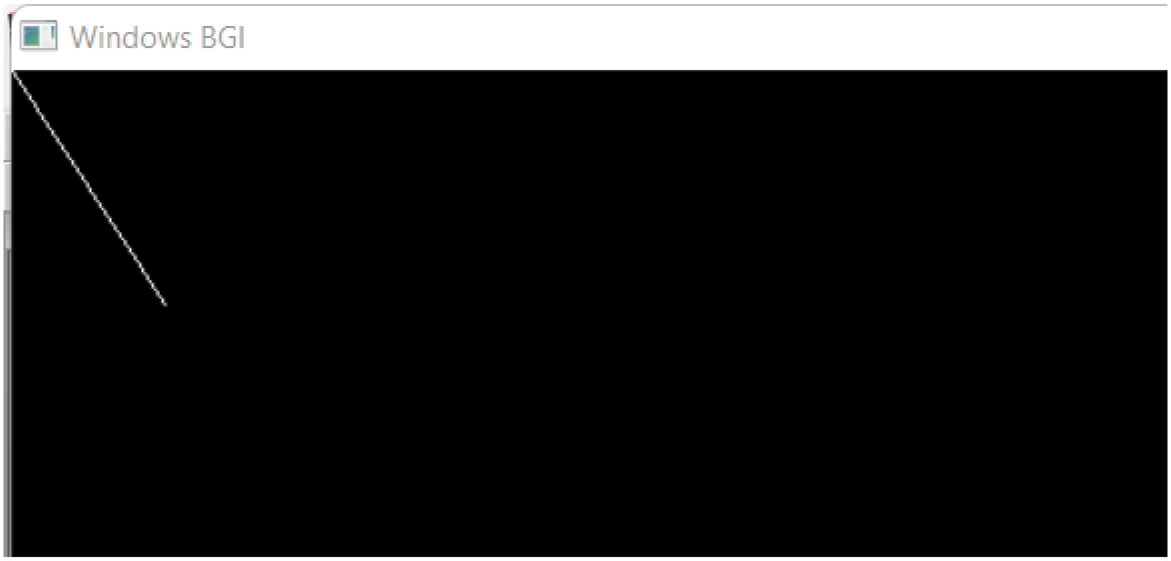
Problem 1: *Rotating canned line at 30 degree angle*

Source Code:

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

void drawLine(int x1, int y1, int x2, int y2) {
    int dx = abs(x2 - x1);
    int dy = abs(y2 - y1);
    int sx = (x1 < x2) ? 1 : -1;
    int sy = (y1 < y2) ? 1 : -1;
    int err = dx - dy;

    while (true) {
        putpixel(x1, y1, WHITE);

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

    int x1 = 0;     // Starting X-coordinate
    int y1 = 0;     // Starting Y-coordinate
    int x2 = 100;   // Ending X-coordinate

    int y2 = 50;     // Ending Y-coordinate

    // Rotate the line by 30 degrees
    double angle = 30 * (3.14159 / 180.0);
    int rotatedX2 = static_cast<int>(x1 + (x2 - x1) *
cos(angle) - (y2 - y1) * sin(angle));
    int rotatedY2 = static_cast<int>(y1 + (x2 - x1) *
  sin(angle) + (y2 - y1) * cos(angle));

    drawLine(x1, y1, rotatedX2, rotatedY2);

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

Output:



Problem 02: *Scaling scanned line at 50%*

Source Code:

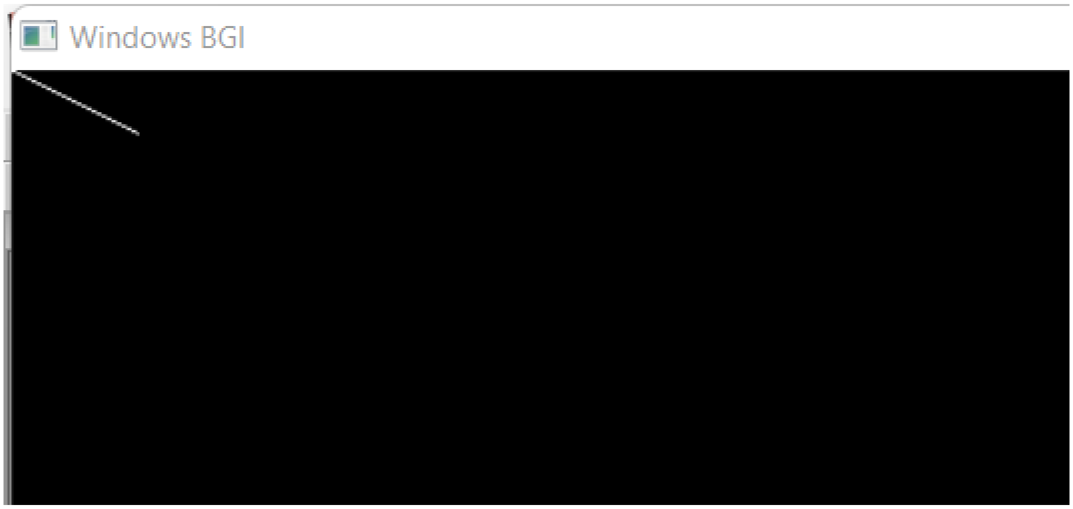
```
#include <graphics.h>

void drawLine(int x1, int y1, int x2, int y2) {
   int dx = abs(x2 - x1);
   int dy = abs(y2 - y1);
   int sx = (x1 < x2) ? 1 : -1;
   int sy = (y1 < y2) ? 1 : -1;
   int err = dx - dy;

   while (true) {
      putpixel(x1, y1, WHITE);

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

      int e2 = 2 * err;

      if (e2 > -dy) {
```

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

   int x1 = 0;     // Starting X-coordinate
   int y1 = 0;     // Starting Y-coordinate
   int x2 = 100;   // Ending X-coordinate
   int y2 = 50;     // Ending Y-coordinate

   // Scale the line by 50%
   double scaleFactor = 0.5;
   int scaledX2 = static_cast<int>(x1 + (x2 - x1) *
scaleFactor);
   int scaledY2 = static_cast<int>(y1 + (y2 - y1) *
scaleFactor);

   drawLine(x1, y1, scaledX2, scaledY2);
```

```
            err -= dy;
            x1 += sx;
        }

        if (e2 < dx) {
            err += dx;
            y1 += sy;
        }
    }
}
```

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

Output:



Problem 03: *Translate scanned line at 75 pixel*

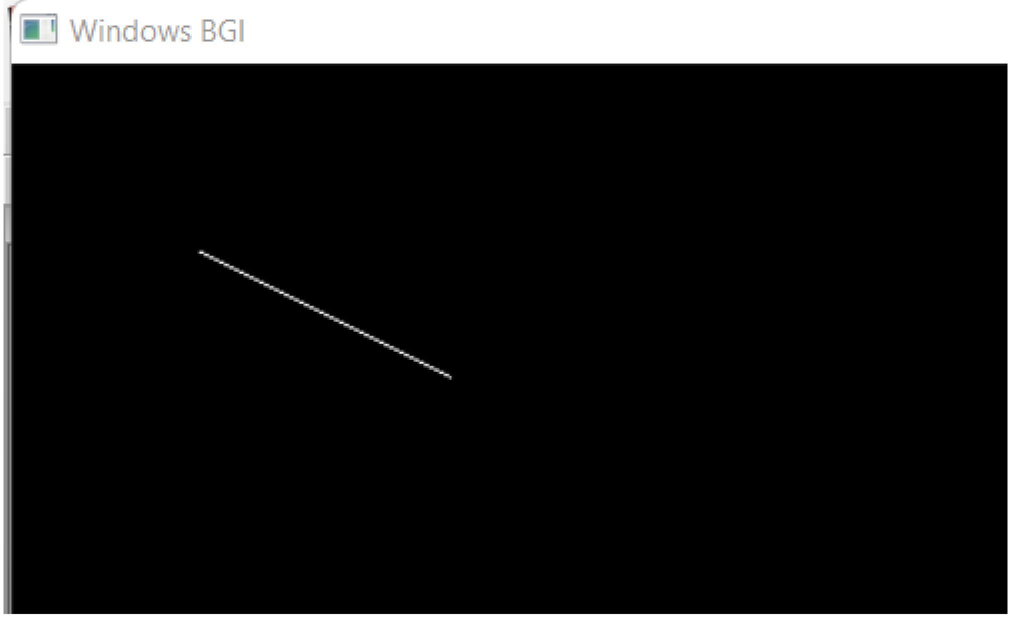
Source Code:

```
#include <graphics.h>

void drawLine(int x1, int y1, int x2, int y2) {
    int dx = abs(x2 - x1);
    int dy = abs(y2 - y1);
```

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

    int x1 = 0;     // Starting X-coordinate
    int sx = (x1 < x2) ? 1 : -1;
    int sy = (y1 < y2) ? 1 : -1;
    int err = dx - dy;

    while (true) {
        putpixel(x1, y1, WHITE);

        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 y1 = 0;     // Starting Y-coordinate
    int x2 = 100;   // Ending X-coordinate
    int y2 = 50;     // Ending Y-coordinate

    // Translate the line by 75 pixels in both x and y directions
    int translatedX1 = x1 + 75;
    int translatedY1 = y1 + 75;
    int translatedX2 = x2 + 75;
    int translatedY2 = y2 + 75;

    drawLine(translatedX1, translatedY1, translatedX2, translatedY2);

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

Output:



Experiment No:02

Name OF Experiment: *Scan Conversion A Kite using Bresenham’s Line algorithm*

Source Code:

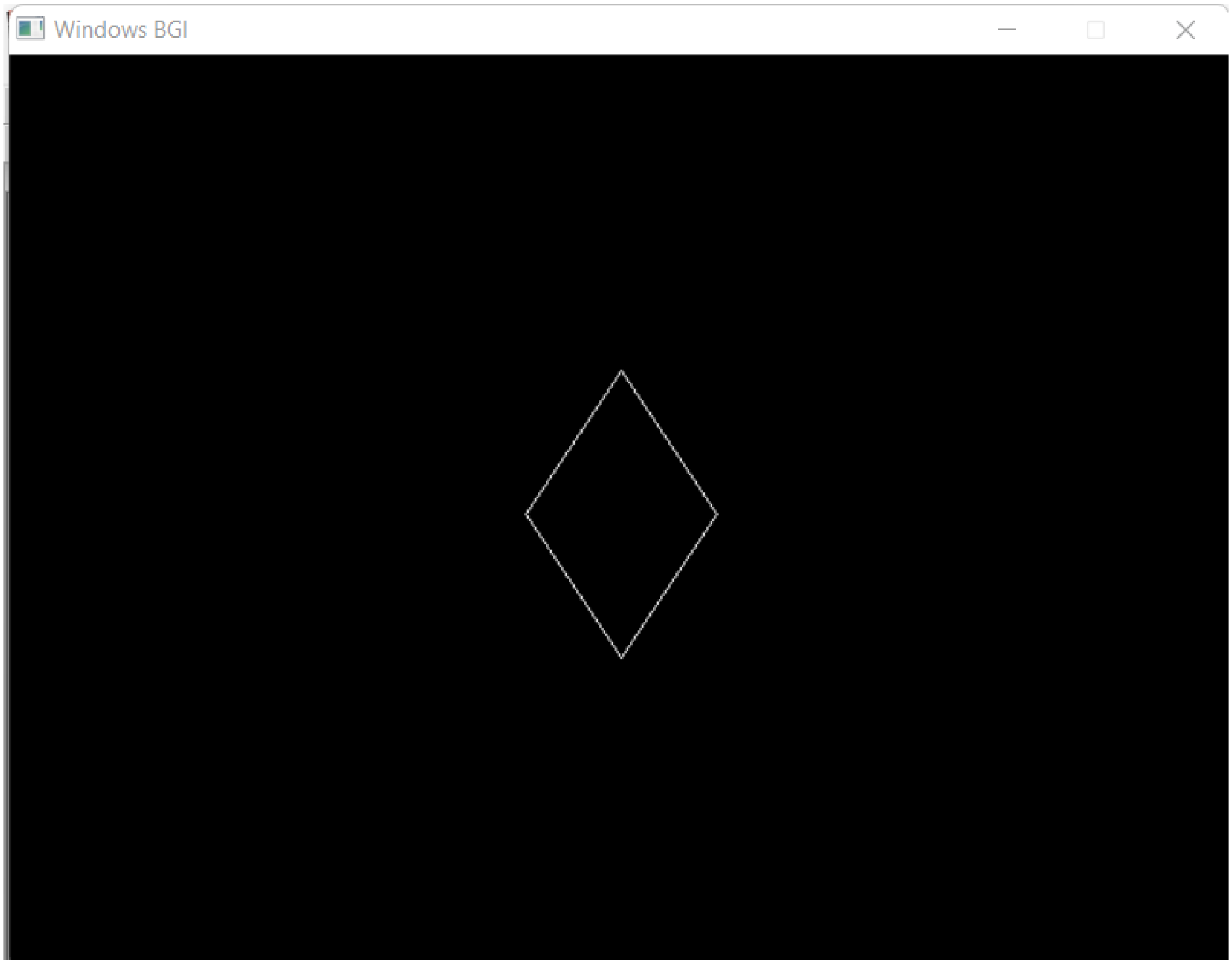
```
#include <graphics.h>

void drawLine(int x1, int y1, int x2, int y2) {
    int dx = abs(x2 - x1);
    int dy = abs(y2 - y1);
    int sx = (x1 < x2) ? 1 : -1;
    int sy = (y1 < y2) ? 1 : -1;
    int err = dx - dy;

    while (true) {
        putpixel(x1, y1, WHITE);

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

        int e2 = 2 * err;

        if (e2 > -dy) {
```

```
 int bottomX = centerX;
    int bottomY = centerY + (height / 2);
    int leftX = centerX - (width / 2);
    int leftY = centerY;
    int rightX = centerX + (width / 2);
    int rightY = centerY;

    drawLine(topX, topY, leftX, leftY);     // Top to
left
    drawLine(topX, topY, rightX, rightY);   // Top to
right
    drawLine(bottomX, bottomY, leftX, leftY);   //
Bottom to left
    drawLine(bottomX, bottomY, rightX, rightY); //
Bottom to right
}

int main() {
```

```
            err -= dy;
            x1 += sx;
        }

        if (e2 < dx) {
            err += dx;
            y1 += sy;
        }
    }
}

void drawKite() {
    int centerX = 320;  // X-coordinate of the kite's
center
    int centerY = 240;  // Y-coordinate of the kite's
center
    int width = 100;    // Width of the kite
    int height = 150;   // Height of the kite

    int topX = centerX;
    int topY = centerY - (height / 2);
```

```
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");

    drawKite();

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

Output:


Windows BGI