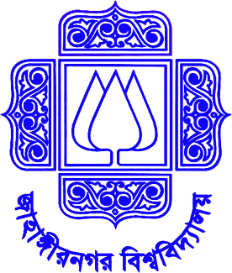
**Lab Report-02**

*Course title: Computer Graphics*

*Course code: CSE-304*

*3rd Year 1st Semester Examination 2022*

**Date of Submission**:04-06-2023

****

**Submitted to-**

###### **Dr Mohammad Shorif Uddin**

*Professor*

*Department of Computer Science and Engineering*

*Jahangirnagar University*

*Savar, Dhaka-1342*

*&*

**Dr Morium Akter**

*Assistant Professor*

*Department of Computer Science and Engineering*

*Jahangirnagar University*

*Savar, Dhaka-1342*

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl** | Class Roll | Exam Roll | Name |
| 01 | 397 |  | Ali Jakey Shariar |

**Problem 1: Scan conversion of a point.**

**Source Code:**

#include<graphics.h>

#include<conio.h>

#include<bits/stdc++.h>

using namespace std;

int main()

{

int gd = DETECT, gm;

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

float x = 200, y = 200;

putpixel (x, y, WHITE);

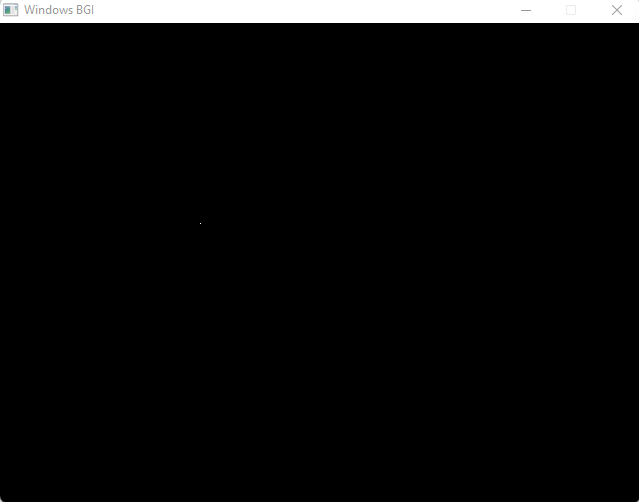
getch();

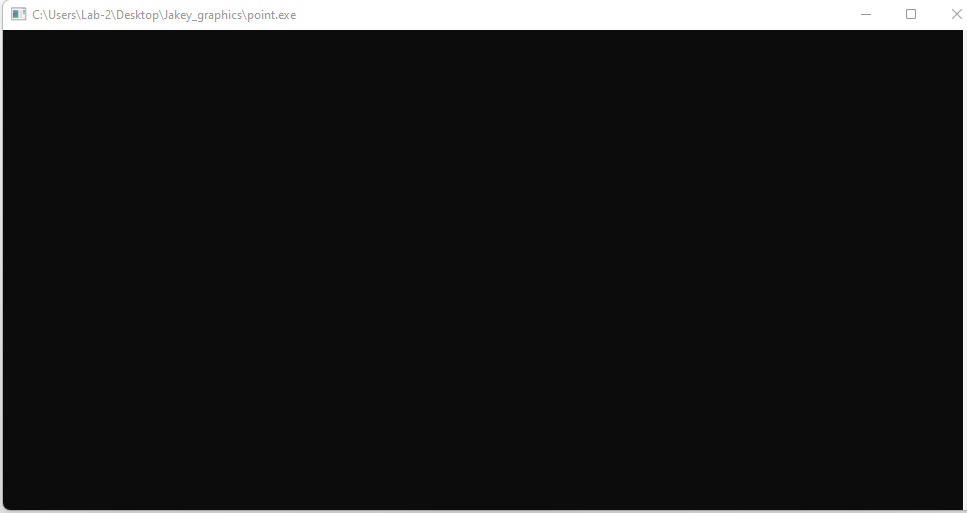
closegraph();

return 0;

}

**Output:**





**Problem 2: Scan Conversion of a line using DDA Algorithm**

**Source Code:**

#include<graphics.h>

#include<conio.h>

#include<stdio.h>

int main()

{

int gd = DETECT ,gm, i;

float x, y,dx,dy,steps;

int x0, x1, y0, y1;

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

setbkcolor(WHITE);

x0 = 100 , y0 = 200, x1 = 500, y1 = 300;

dx = (float)(x1 - x0);

dy = (float)(y1 - y0);

if(dx>=dy)

{

steps = dx;

}

else

{

steps = dy;

}

dx = dx/steps;

dy = dy/steps;

x = x0;

y = y0;

i = 1;

while(i<= steps)

{

putpixel(x, y, WHITE);

x += dx;

y += dy;

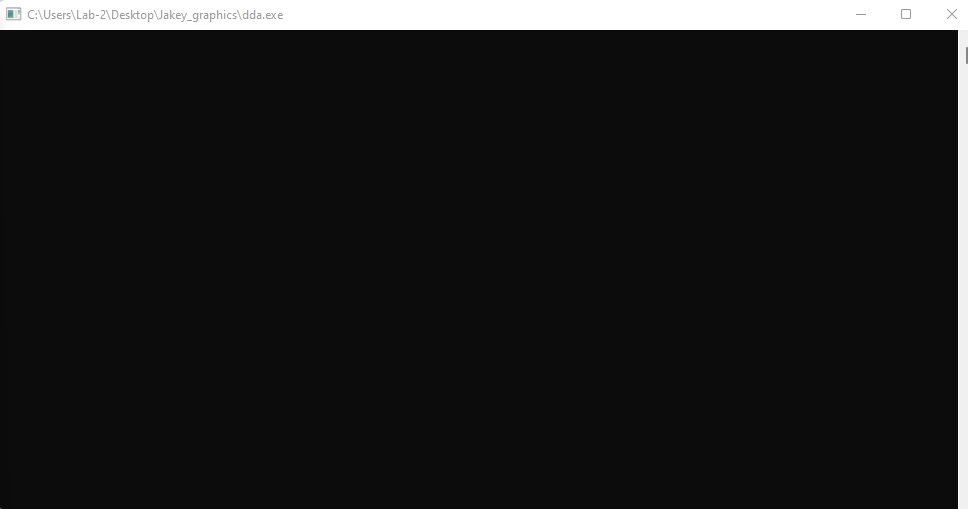
i=i+1;

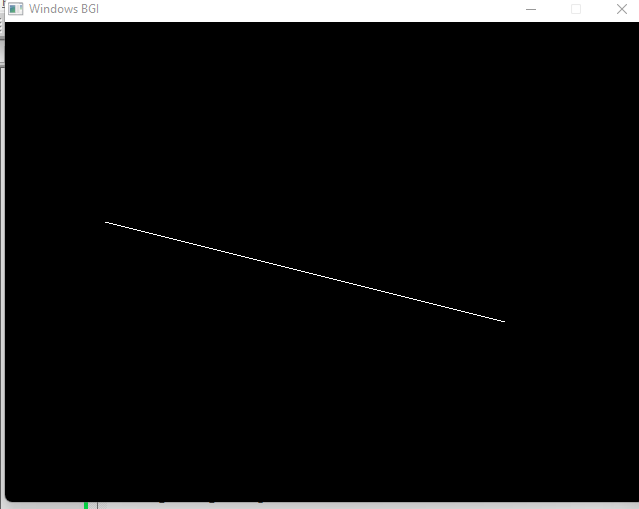
}

getch();

closegraph();

}

**Output: **

****

**Problem 3: Scan Conversion of line using Bresenham Algorithm**

**Source Code:**

#include<stdio.h>

#include<graphics.h>

#include<bits/stdc++.h>

void drawline(int x0, int y0, int x1, int y1)

{

int dx, dy, p, x, y;

dx=x1-x0;

dy=y1-y0;

x=x0;

y=y0;

p=2\*dy-dx;

while(x<x1)

{

if(p>=0)

{

putpixel(x,y,7);

y=y+1;

p=p+2\*dy-2\*dx;

}

else

{

putpixel(x,y,7);

p=p+2\*dy;}

x=x+1;

}

}

int main()

{

int gd = DETECT, gm;

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

int error, x0, y0, x1, y1;

// initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");

printf("Enter co-ordinates of first point: ");

scanf("%d%d", &x0, &y0);

printf("Enter co-ordinates of second point: ");

scanf("%d%d", &x1, &y1);

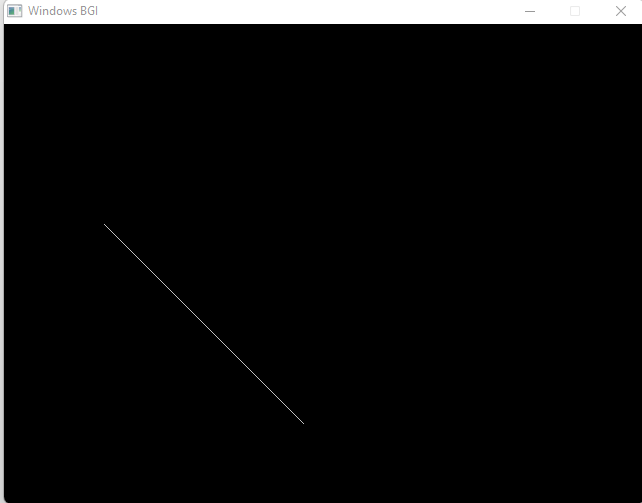
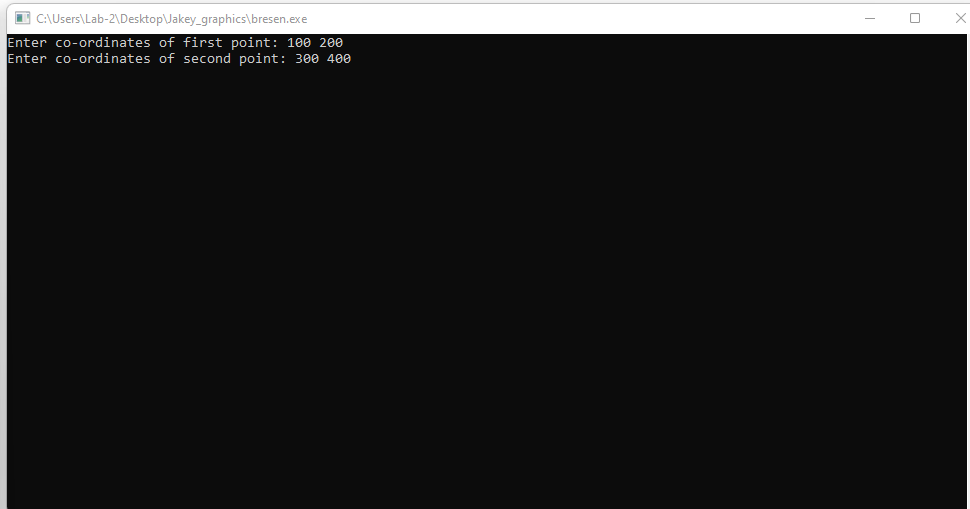
drawline(x0, y0, x1, y1);

getch();

return 0;

}

**Output:**

****

**Problem 4: Scan Conversion of circle using bresenhen Algorithm**

**Source Code:**

#include <graphics.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <math.h>

void EightWaySymmetricPlot(int xc,int yc,int x,int y)

{

putpixel(x+xc,y+yc,RED);

putpixel(x+xc,-y+yc,YELLOW);

putpixel(-x+xc,-y+yc,GREEN);

putpixel(-x+xc,y+yc,YELLOW);

putpixel(y+xc,x+yc,12);

putpixel(y+xc,-x+yc,14);

putpixel(-y+xc,-x+yc,15);

putpixel(-y+xc,x+yc,6);

}

void BresenhamCircle(int xc,int yc,int r)

{

int x=0,y=r,d=3-(2\*r);

EightWaySymmetricPlot(xc,yc,x,y);

while(x<=y)

{

if(d<=0)

{

d=d+(4\*x)+6;

}

else

{

d=d+(4\*x)-(4\*y)+10;

y=y-1;

}

x=x+1;

EightWaySymmetricPlot(xc,yc,x,y);

}

}

int main(void)

{

/\* request auto detection \*/

int xc,yc,r,gdriver = DETECT, gmode, errorcode;

/\* initialize graphics and local variables \*/

initgraph(&gdriver, &gmode, "C:\\TURBOC3\\BGI");

/\* read result of initialization \*/

errorcode = graphresult();

if (errorcode != grOk) /\* an error occurred \*/

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1); /\* terminate with an error code \*/

}

printf("Enter the values of xc and yc :");

scanf("%d%d",&xc,&yc);

printf("Enter the value of radius :");

scanf("%d",&r);

BresenhamCircle(xc,yc,r);

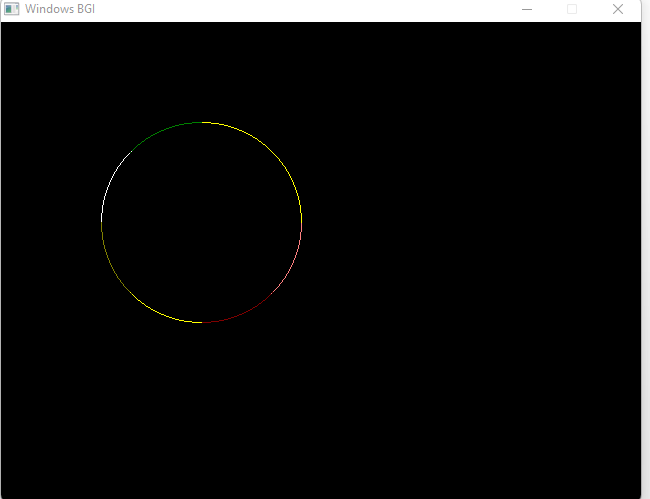
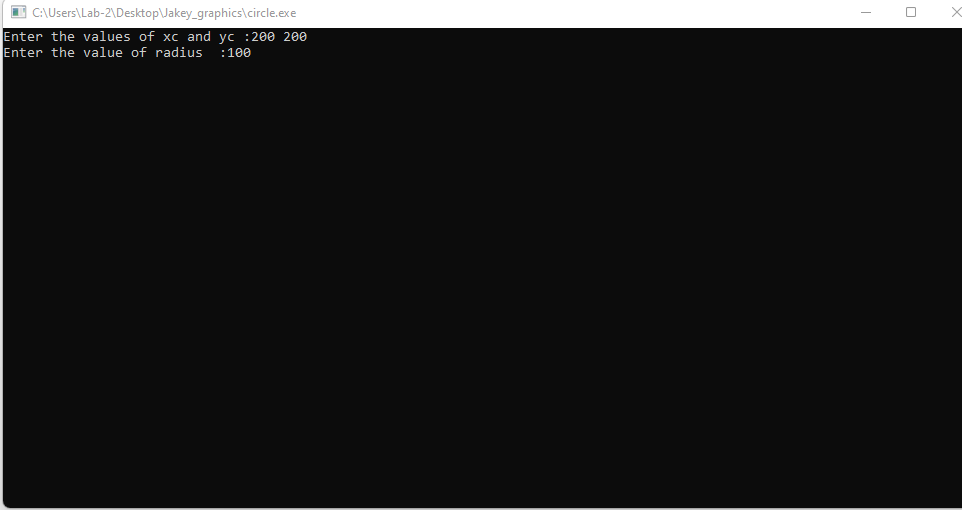
getch();

closegraph();

return 0;

}

**Output:**

****

**Problem 5: Midpoint**

**Source Code:**

#include <iostream>

#include <graphics.h>

// Function to plot points in all octants

void plotPoints(int xc, int yc, 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);

}

// Midpoint Circle Drawing Algorithm

void midpointCircle(int xc, int yc, int r)

{

int x = 0;

int y = r;

int p = 1 - r;

// Plot the first point

plotPoints(xc, yc, x, y);

while (x < y)

{

x++;

if (p < 0)

p += 2 \* x + 1;

else

{

y--;

p += 2 \* (x - y) + 1;

}

// Plot points in all octants

plotPoints(xc, yc, x, y);

}

}

int main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "");

int xc = 320; // X-coordinate of the center

int yc = 240; // Y-coordinate of the center

int r = 100; // Radius of the circle

midpointCircle(xc, yc, r);

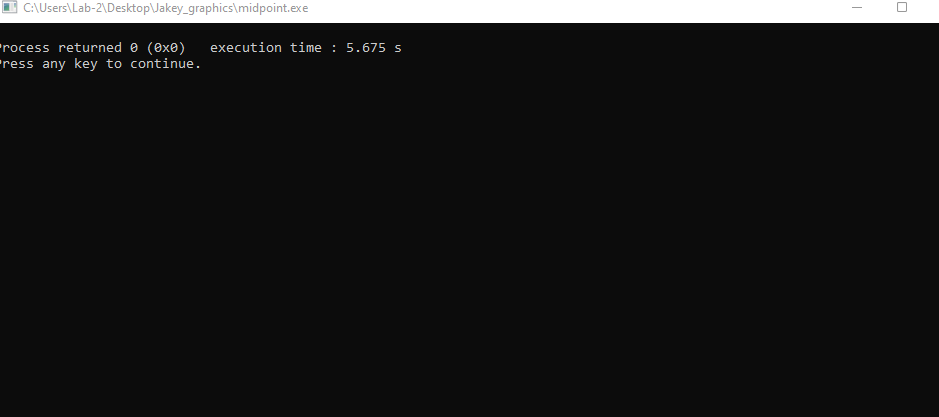
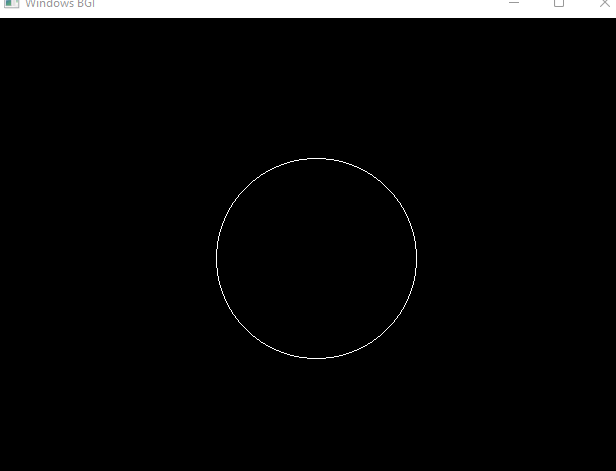
delay(5000);

closegraph();

return 0;

}

**Output:**

****

**Problem: 6:** **Ellipse**

**Source Code:**

#include <iostream>

#include <graphics.h>

#include <conio.h>

// Function to draw ellipse using midpoint algorithm

void drawEllipse(int xc, int yc, int rx, int ry)

{

int x, y;

int p;

int px, py;

// Initialize starting point

x = 0;

y = ry;

// Calculate initial decision parameter

p = (ry \* ry) - (rx \* rx \* ry) + ((rx \* rx) / 4);

// Plot the points of first region

while ((2 \* x \* ry \* ry) < (2 \* y \* rx \* rx))

{

// Plot points in Region 1

putpixel(xc + x, yc - y, WHITE);

putpixel(xc - x, yc + y, WHITE);

putpixel(xc + x, yc + y, WHITE);

putpixel(xc - x, yc - y, WHITE);

// Checking and updating decision parameter

if (p < 0)

{

x++;

px = 2 \* ry \* ry \* x;

p = p + (px + ry \* ry);

}

else

{

x++;

y--;

px = 2 \* ry \* ry \* x;

py = 2 \* rx \* rx \* y;

p = p + (px - py + ry \* ry);

}

}

// Calculate decision parameter for the second region

p = ((ry \* ry) \* (x + 0.5) \* (x + 0.5)) +

((rx \* rx) \* (y - 1) \* (y - 1)) - (rx \* rx \* ry \* ry);

// Plot the points of the second region

while (y >= 0)

{

// Plot points in Region 2

putpixel(xc + x, yc - y, WHITE);

putpixel(xc - x, yc + y, WHITE);

putpixel(xc + x, yc + y, WHITE);

putpixel(xc - x, yc - y, WHITE);

// Checking and updating decision parameter

if (p > 0)

{

y--;

py = 2 \* rx \* rx \* y;

p = p + (rx \* rx - py);

}

else

{

y--;

x++;

px = 2 \* ry \* ry \* x;

py = 2 \* rx \* rx \* y;

p = p + (px - py + rx \* rx);

}

}

}

int main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, "");

// Set the center and radii of the ellipse

int centerX = 250;

int centerY = 250;

int radiusX = 100;

int radiusY = 50;

// Call the function to draw the ellipse

drawEllipse(centerX, centerY, radiusX, radiusY);

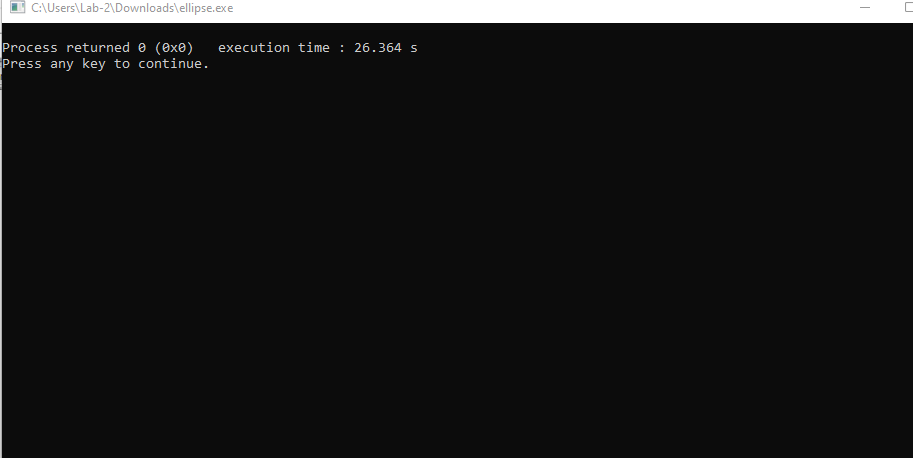
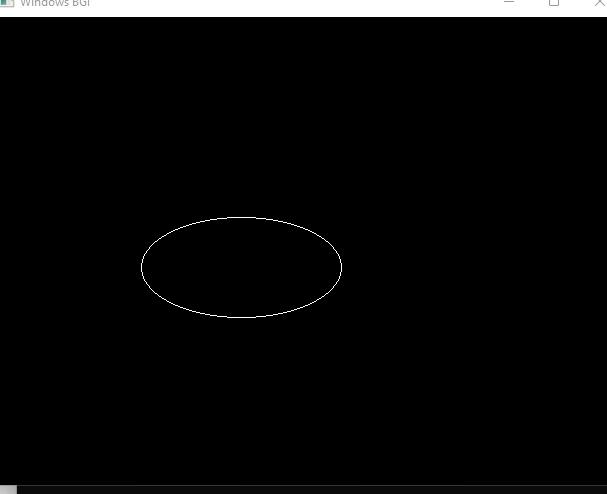
getch();

closegraph();

return 0;

}

**Output:**

****