<u>ASSIGNMENT NO :</u> 1

TITLE:

Implementation DDA Line Drawing algorithm

OBJECTIVE :

To get the fundamental concept for line drawing technique in computer graphics

THEORY :

In Computer Graphics the first basic line drawing algorithm is Digital Differential Analyzer (DDA) Algorithm.

DDA line drawing algorithm is a scan conversion line drawing algorithm based on calculating either dx or dy from the equation

$$m=dy/dx=(y2-y1)/(x2-x1)$$

In DDA algorithm, we sample the line at unit intervals in one coordinate and determine the corresponding integer value nearest to the line path for the other coordinates.

ALGORITHM :

```
Step 1: Read the input of the 2 end points of the line as (x1, y1) & (x2, y2) such that
        x1 != x2 and y1 != y2
Step 2: Calculate dx = x2 – x1 and dy = y2 – y1
Step 3: if(dx>=dy)
            step=dx
        otherwise,
            step=dy
Step 4: calculate, xin = dx / step & yin = dy / step
Step 5: set ,x = x1 + 0.5 & y = y1 + 0.5
Step 6: for(k = 0; k < step; k++) do
            i.   x = x + xin
            ii.  y = y + yin
            iii. putpixel(x, y)
```

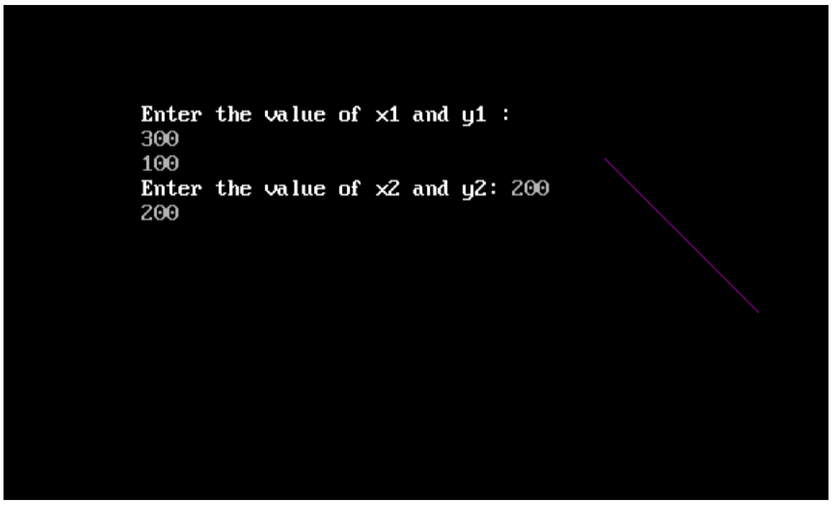
PROBLEM TITLE :

Write a program to implement DDA Line Drawing Algorithm in C to draw a line between given points

SOURCE CODE :

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

void main( )
{
        float x,y,x1,y1,x2,y2,dx,dy,step;
        int i,gd=DETECT,gm;

        initgraph(&gd,&gm,"c:\\turboc3\\bgi");

        printf("Enter the value of x1 and y1 : ");
        scanf("%f%f",&x1,&y1);
        printf("Enter the value of x2 and y2: ");
        scanf("%f%f",&x2,&y2);

        dx=abs(x2-x1);
        dy=abs(y2-y1);

        if(dx>=dy)
                step=dx;
        else
                step=dy;

        dx=dx/step;
        dy=dy/step;
        x=x1;
        y=y1;
        i=1;
        while(i<=step)
        {
                putpixel(x,y,5);
                x=x+dx;
                y=y+dy;
                i=i+1;
                delay(100);
        }
          getch();
        closegraph();
}
```

OUTPUT :



```
Enter the value of x1 and y1 :
300
100
Enter the value of x2 and y2: 200
200
```

ASSIGNMENT NO : 2

TITLE:

Implementation BLA Line Drawing algorithm

OBJECTIVE :

To get the fundamental concept for line drawing technique in computer graphics.

THEORY:

Bresenham's Line Drawing Algorithm (BLA) is another incremental scan conversion algorithm. The big advantage of the algorithm is that it uses only integer calculation. And the integer calculation is made by a parameter called decision parameter. That is used to determine the pixel position of next plot.

ALGORITHM :

```
Step 1 : start
Step 2 : Input two end points of the line, left end point in (x0,y0) and right end point in
(x1,y1)
Step 3 : Calculate dx = abs(x1 –x 0)
                    Dy = abs(y1-y0)
Step 4 : Compute initial decision parameter
                P= 2 dy – dx
Step 5 : if  (x1 > x0) then
                    i.    Set, x=x0
                   ii.    y = y0
                  iii.    xend = x1
         Otherwise,
                    i.    x =x1
                   ii.    y =y1
                  iii.    xend = x0
Step 6 : plotpixel (x,y)
Step 7 : While (x < xend)
                i.    x ++
               ii.    if  ( p < 0)
                        a)   p = p + 2 dy
              iii.    otheriwise ,
                        a)    y ++
                        b)  p  = p + 2 dy – 2 dx
                        c)  plotpixel ( x , y)
```

Step 8 : stop

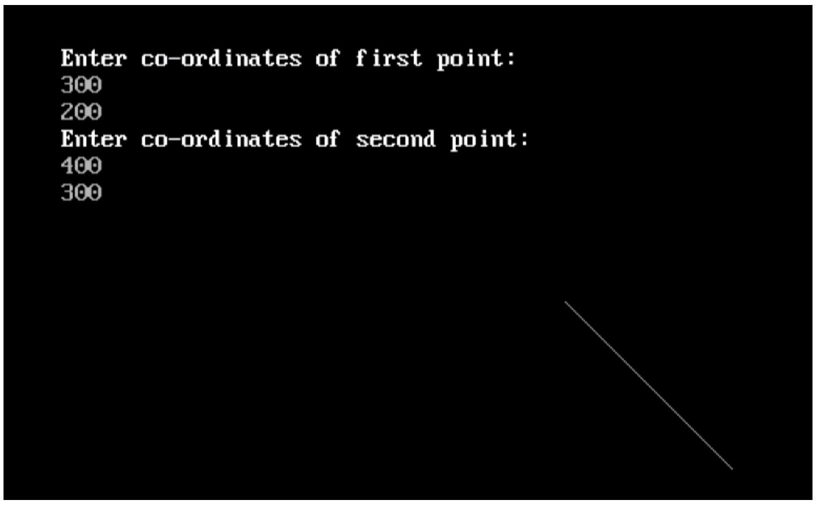
PROBLEM TITLE :

Write a program to implement BLA Line Drawing Algorithm in C

SOURCE CODE :

```
#include<stdio.h>
#include <conio.h>
#include<graphics.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 gdriver=DETECT, gmode, 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:



```
Enter co-ordinates of first point:
300
200
Enter co-ordinates of second point:
400
300
```

ASSIGNMENT NO : 3

TITLE:

Implementation of Midpoint circle algorithm

OBJECTIVE:

To get the fundamental concept for circle drawing technique in computer graphics.

THEORY:

In midpoint circle algorithm, we sample at unit intervals and determine the closest pixel position to the specified circle path at each step. For a given radius ‘r’ and screen center position (Xc , Yc) we can set up our algorithm to calculate pixel centered at (0 ,0) and then each calculated pixel position (X ,Y) is moved to its proper position by adding ‘Xc’ to ‘X’ and ‘Yc’ to ‘Y’.

$$X = X + Xc$$

$$Y = Y + Yc$$

It uses the symmetric property of circle to reduce computational time. Pixel positions are only calculated within an octant and transfer them through remaining seven octant to get a complete circle.

ALGORITHM :

```
Step 1 : start
Step 2 : Input  radius ‘r’ and center point of the circle ( h,k)
Step 3 : Initilize , X =0 and Y= r and P = 1 - r
Step 4 : while ( X <= Y)
            a.  plotpixel ( X + h, Y + k)
                plotpixel (- X + h, Y + k)
                plotpixel ( X + h,- Y + k)
                plotpixel (- X + h, -Y + k)
                plotpixel ( Y + h, X + k)
                plotpixel ( -Y + h, X + k)
                plotpixel ( Y + h,- X + k)
                plotpixel ( -Y + h,- X + k)
            b.  if  ( P < 0) then
```

```
                set ,X = X +1
                     P = P + 2 X + 1

          c.  Otherwise ,
                set ,  x = x +1
                      Y = y -1
                P = P +2X+1-2Y
```

Step 5 : end while

Step 6 : stop

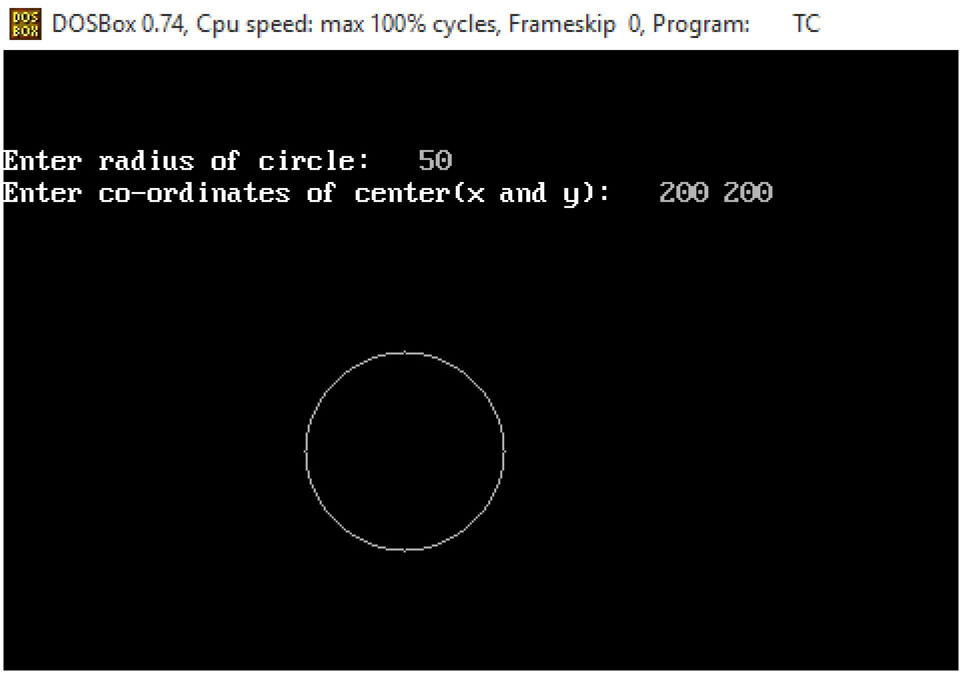
PROBLEM TITLE :

Write a program to implement Bresenham’s Midpoint Circle Drawing Algorithm in C

SOURCE CODE :

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

void drawcircle(int x0, int y0, int radius)
{
   int y = radius;
   int x = 0;
   int p=1-radius;

   while (x <= y)
   {
putpixel(x0 + x, y0 + y, 7);
putpixel(x0 + y, y0 + x, 7);
putpixel(x0 - y, y0 + x, 7);
putpixel(x0 - x, y0 + y, 7);
putpixel(x0 - x, y0 - y, 7);
putpixel(x0 - y, y0 - x, 7);
putpixel(x0 + y, y0 - x, 7);
putpixel(x0 + x, y0 - y, 7);
if (p <= 0)
{
x++;
 p=p+2*x+1;
}
if (p > 0)
{
x++;
y--;
p=p+2*x-2*y+1;
}
    }
}
int main()
{
int gdriver=DETECT, gmode, p, x, y, r;
initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");
printf("\n\n\nEnter radius of circle: ");
scanf("%d", &r);
printf("Enter co-ordinates of center(x and y): ");
scanf("%d%d", &x, &y);
drawcircle(x, y, r);
return 0;
}
```

OUTPUT :



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

```
Enter radius of circle:   50
Enter co-ordinates of center(x and y):   200 200
```

ASSIGNMENT NO: 4

TITLE:

Implementation of Midpoint Ellipse drawing algorithm

OBJECTIVE:

To get the fundamental concept for ellipse drawing technique in computer graphics

THEORY:

The midpoint ellipse algorithm decides which point near the boundary is closure to the actual ellipse path described by the ellipse equation. The point is taken as next point.

Let us consider one quarter of an ellipse. The curve (quarter ellipse) is divided into two regions – region-I and region-II. In region-I, the slope of the curve is greater than -1 while in region-II the slope is less that -1.

PROBLEM TITLE:

Write a program to implement Midpoint Ellipse Drawing Algorithm in C

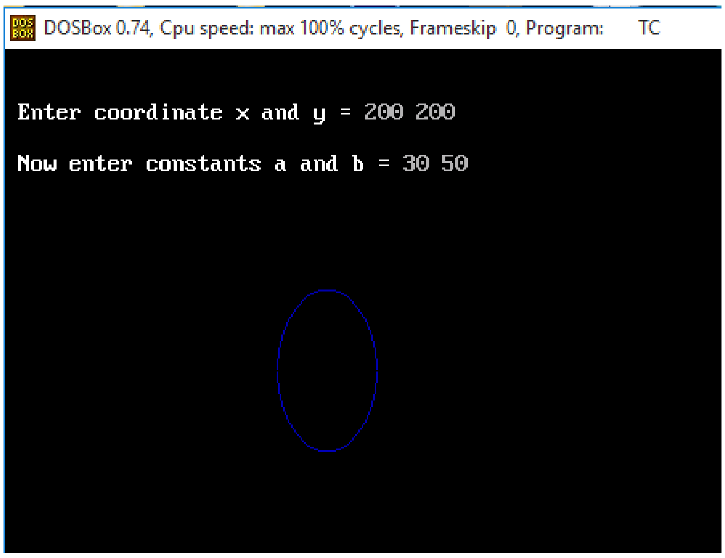
SOURCE CODE:

```
#include<stdio.h>
#include <conio.h>
#include<graphics.h>
void main(){
        long x,y,x_center,y_center;
        long a_sqr,b_sqr, fx,fy, d,a,b,tmp1,tmp2;
        int g_driver=DETECT,g_mode;
        clrscr();
       initgraph(&g_driver,&g_mode,"C:\\TURBOC3\\BGI");

       printf("\n\n Enter coordinate x and y = ");
       scanf("%ld%ld",&x_center,&y_center);
       printf("\n Now enter constants a and b = ");
       scanf("%ld%ld",&a,&b);
       x=0;
       y=b;
       a_sqr=a*a;
       b_sqr=b*b;
       fx=2*b_sqr*x;
       fy=2*a_sqr*y;
 d=b_sqr-(a_sqr*b)+(a_sqr*0.25);
 do
```

```
{
        putpixel(x_center+x,y_center+y,1);
        putpixel(x_center-x,y_center-y,1);
        putpixel(x_center+x,y_center-y,1);
        putpixel(x_center-x,y_center+y,1);

if(d<0)
 {
        d=d+fx+b_sqr;
 }
else
        {
        y=y-1;
        d=d+fx+-fy+b_sqr;
        fy=fy-(2*a_sqr);
        }
        x=x+1;
        fx=fx+(2*b_sqr);
        delay(10);

}
while(fx<fy);
         tmp1=(x+0.5)*(x+0.5);
         tmp2=(y-1)*(y-1);
         d=b_sqr*tmp1+a_sqr*tmp2-(a_sqr*b_sqr);
do
{
        putpixel(x_center+x,y_center+y,1);
        putpixel(x_center-x,y_center-y,1);
        putpixel(x_center+x,y_center-y,1);
        putpixel(x_center-x,y_center+y,1);

if(d>=0)
        d=d-fy+a_sqr;
else

        {
        x=x+1;
        d=d+fx-fy+a_sqr;
        fx=fx+(2*b_sqr);
        }
         y=y-1;
         fy=fy-(2*a_sqr);
```

```
  }
  while(y>0);
  getch();
  closegraph();
}
```

OUTPUT:



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

```
Enter coordinate x and y = 200 200

Now enter constants a and b = 30 50
```

<u>ASSIGNMENT NO</u>: 5

TITLE:

The use of GLUT to draw polygon

OBJECTIVE:

To get familiar with GLUT - The OpenGL Utility Toolkit .

THEORY:

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres and the Utah teapot. GLUT also has some limited support for creating pop-up menus.

The two aims of GLUT are to allow the creation of rather portable code between operating systems (GLUT is cross-platform) and to make learning OpenGL easier. Getting started with OpenGL programming while using GLUT often takes only a few lines of code and does not require knowledge of operating system–specific windowing APIs.

PROBLEM TITLE:

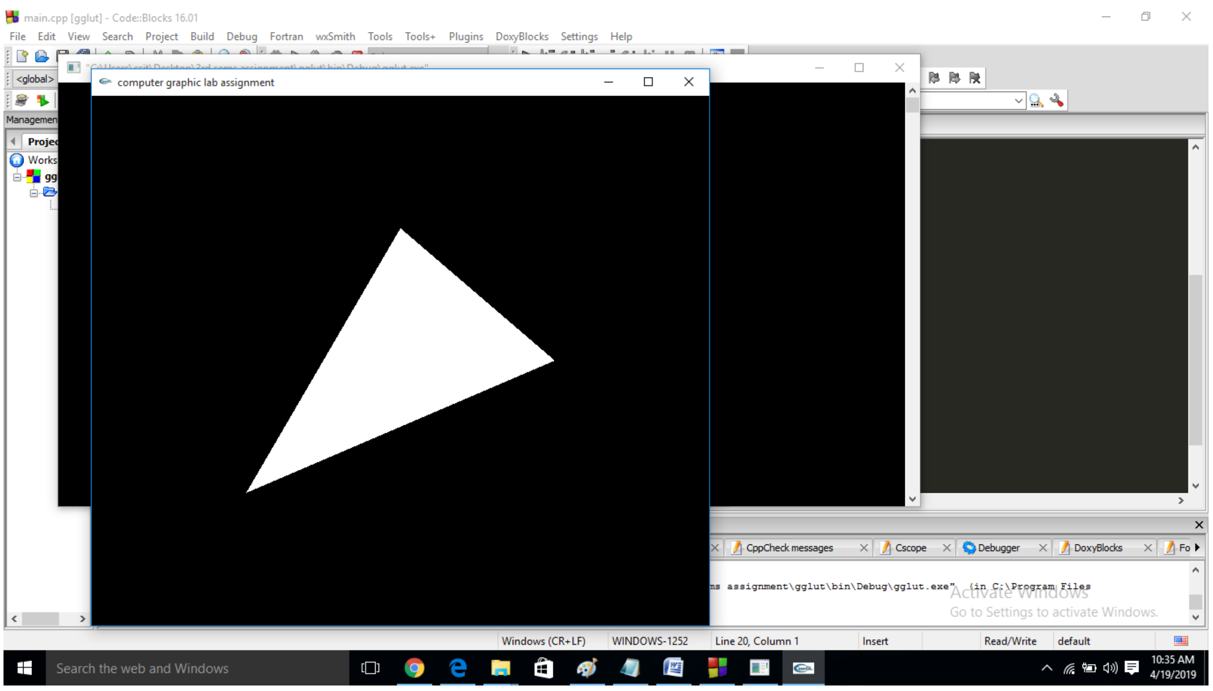
Write a program to draw triangle in GLUT.

SOURCE CODE:

```
#include <windows.h>  // for MS Windows
#include <GL/glut.h>  // GLUT, include glu.h and gl.h
#include <GL/gl.h>
#include <math.h>
void display(void){

   glClear(GL_COLOR_BUFFER_BIT);

 //to create triangle
```

```
    glBegin(GL_TRIANGLES);
        glVertex2f(-0.5,-0.5);
        glVertex2d(0.5,0.0);
        glVertex2d(0.0,0.5);

    glEnd();
   glutSwapBuffers();
}

int main(int argc,char ** argv){
 glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGB);
 glutInitWindowPosition(100,100);
 glutInitWindowSize(700,600);
 glutCreateWindow("computer graphic lab assignment ");
 glutDisplayFunc(display);

 glutMainLoop();

return 0;
}
```

OUTPUT:



main.cpp [gglut] - Code::Blocks 16.01
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
computer graphic lab assignment
CppCheck messages
Cscope
Debugger
DoxyBlocks
Windows (CR+LF)
WINDOWS-1252
Line 20, Column 1
Insert
Read/Write
default
Search the web and Windows
10:35 AM
4/19/2019

```
ns assignment\gglut\bin\Debug\gglut.exe"  (in C:\Program Files
```

ASSIGNMENT NO : 6

TITLE:

The Use of GLUT to draw a circle

OBJECTIVE:

To get familiar with GLUT - The OpenGL Utility Toolkit.

THEORY:

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programs, which primarily perform system-level I/O with the host operating system. Functions performed include window definition, window control, and monitoring of keyboard and mouse input. Routines for drawing a number of geometric primitives (both in solid and wireframe mode) are also provided, including cubes, spheres and the Utah teapot. GLUT also has some limited support for creating pop-up menus.

PROBLEM TITLE:

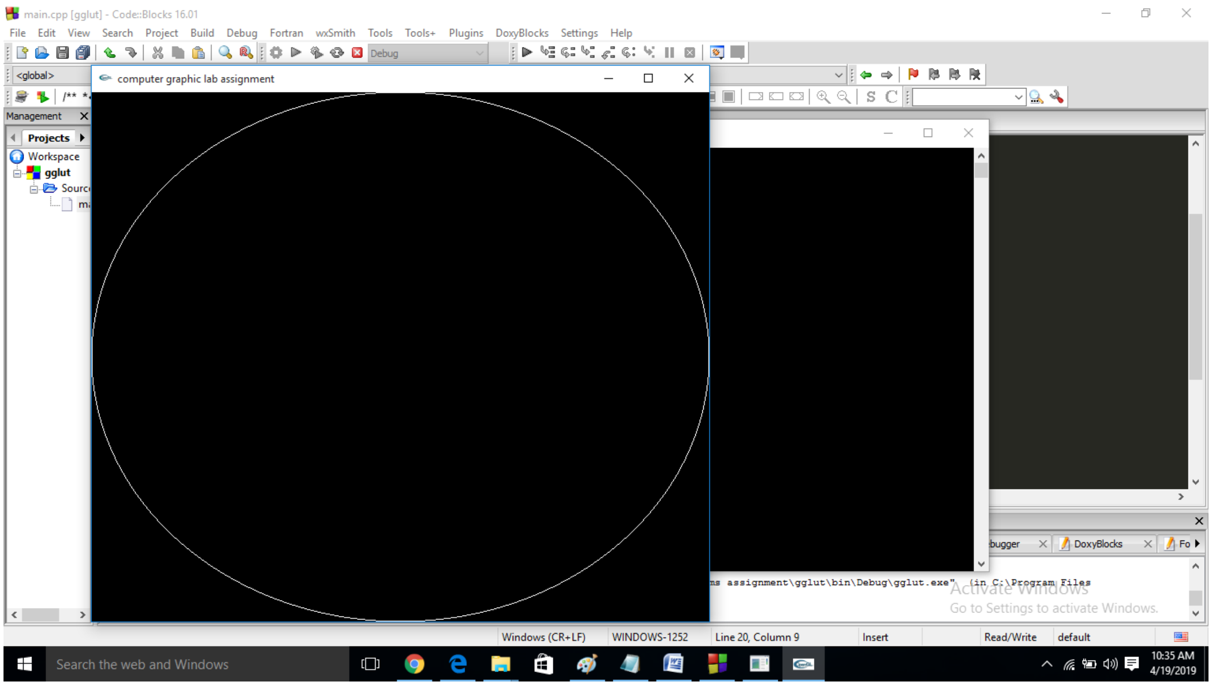
Write a program to create circle in GLUT.

SOURCE CODE:

```
#include <windows.h>  // for MS Windows
#include <GL/glut.h>  // GLUT, include glu.h and gl.h
#include <GL/gl.h>
#include <math.h>
void display(void){
   glClear(GL_COLOR_BUFFER_BIT);
//to create circle
  glBegin(GL_LINE_LOOP);
    for(int i =0; i <= 360; i++){                    // i = angle
        double angle = 3.14 * i / 180;   //converting angle into radian
        double x = cos(angle);
        double y = sin(angle);
        glVertex2d(x,y);
    }
glEnd();
```

```
  glutSwapBuffers();
}

int main(int argc,char ** argv){
 glutInit(&argc,argv);
 glutInitDisplayMode(GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGB);
 glutInitWindowPosition(100,100);
 glutInitWindowSize(700,600);
 glutCreateWindow("computer graphic lab assignment ");
 glutDisplayFunc(display);

 glutMainLoop();
return 0;
}
```

OUTPUT:


```
s assignment\gglut\bin\Debug\gglut.exe"  (in C:\Program Files
```
main.cpp [gglut] - Code::Blocks 16.01
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
computer graphic lab assignment
Management
Projects
Workspace
gglut
Activate Windows
Go to Settings to activate Windows.
Windows (CR+LF) WINDOWS-1252 Line 20, Column 9 Insert Read/Write default
Search the web and Windows
10:35 AM
4/19/2019