COMPUTER GRAPHICS LAB FILE

SUBMITTED BY:

NAME: ZISHNENDU SARKER

ROLL: 2K19/C0/450

SECTION : A6

DATA STRUCTURE LAB (G3)

SUBMITTED TO :

RAJU KUMAR SIR

DEPARTMENT OF COMPUTER SCIENCE &ENGINEERING

VISION

Department of Computer Science & Engineering to be a leading world class technology department playing its role as a key node in national and global knowledge network, thus empowering the computer science industry with the wings of knowledge and power of innovation.

MISSION

The Mission of the department is as follows:

1. To nurture talent of students for research, innovation and excellence in the field of computer engineering starting from Under graduate level.

2. To develop highly analytical and qualified computer engineers by imparting training on cutting edge technology.

3. To produce socially sensitive computer engineers with professional ethics.

4. To focus on R&D environment in close partnership with industry and foreign universities.

5. To produce well-rounded, up to date, scientifically tempered, design-oriented engineer and scientists capable of lifelong learning

INDEX:

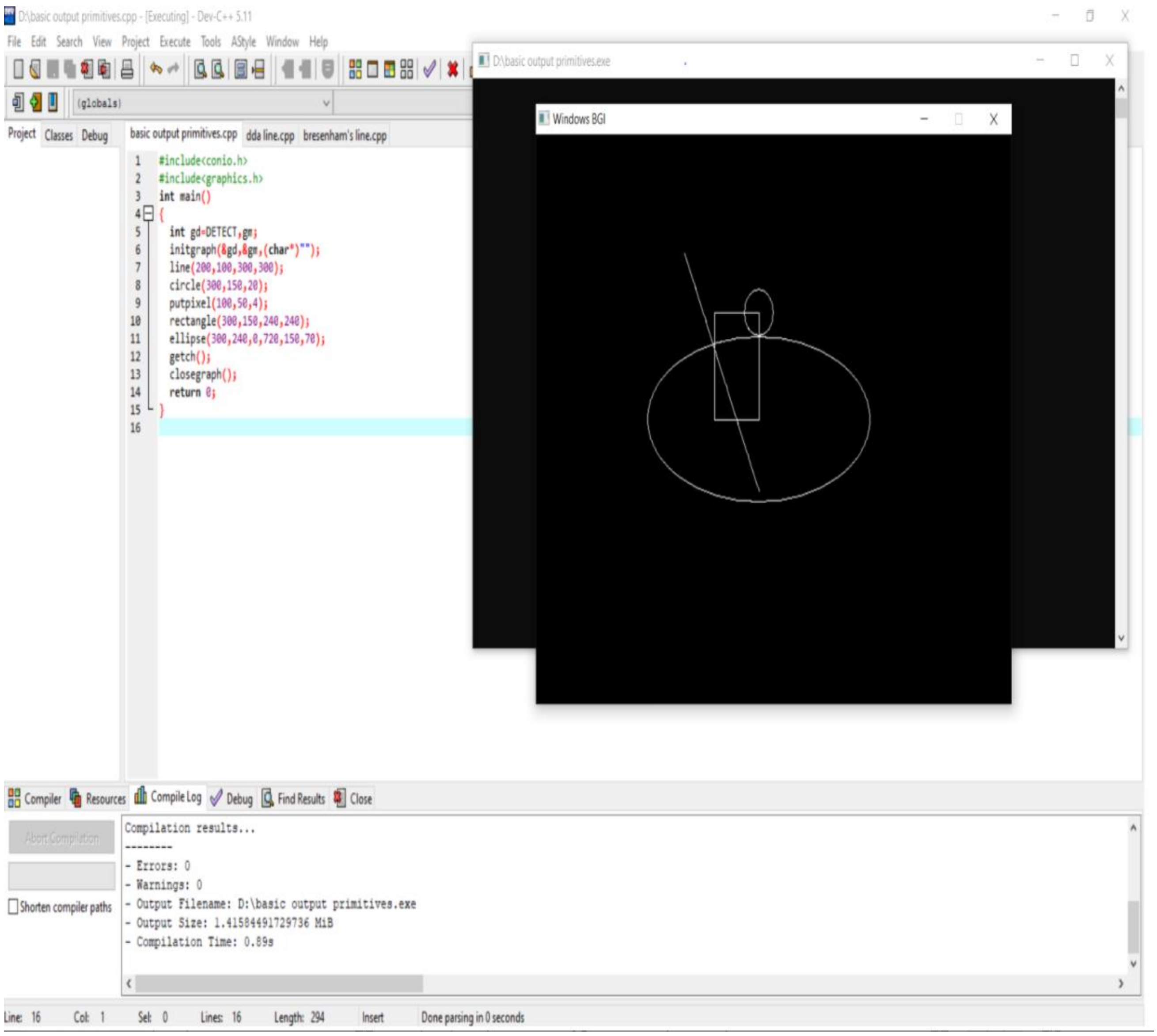
LAB ASSIGNMENT NO. 01

Question :- **Write a program to draw basic output primitives (pixel, line, circle , rectangle, eclipse).**

Code:

```
#include<conio.h>
#include<graphics.h>
int main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,(char*)"");
line(200,100,300,300);
circle(300,150,20);
putpixel(100,50,4);
rectangle(300,150,240,240);
ellipse(300,240,0,720,150,70);
getch();
closegraph();
return 0;
}
```

Output:


D:\basic output primitives.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
basic output primitives.cpp
dda line.cpp
bresenham's line.cpp
D:\basic output primitives.exe
Windows BGI
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 16 Col: 1 Sel: 0 Lines: 16 Length: 294 Insert Done parsing in 0 seconds

```
#include<conio.h>
#include<graphics.h>
int main()
{
  int gd=DETECT,gm;
  initgraph(&gd,&gm,(char*)"");
  line(200,100,300,300);
  circle(300,150,20);
  putpixel(100,50,4);
  rectangle(300,150,240,240);
  ellipse(300,240,0,720,150,70);
  getch();
  closegraph();
  return 0;
}
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\basic output primitives.exe
- Output Size: 1.41584491729736 MiB
- Compilation Time: 0.89s
```

LAB NO. 2

Question : Write a program to implement DDA line drawing algorithm.

Code:

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

```
#include <iostream>
#include <math.h>
#include <conio.h>
int main()
{
float x,y,x1,y1,x2,y2,dx,dy,step;
std::cout<<"X1 is ";
std::cin>>x1;
std::cout<<" and "<<'\n'<<"X2 is ";
std::cin>>x2;
std::cout<<'\n'<<"Y1 is ";
std::cin>>y1;
std::cout<<" and "<<'\n'<<"Y2 is ";
std::cin>>y2;
int gd=DETECT,gm;
initgraph(&gd,&gm,(char*)"");
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;
for(int i=1;i<step;i++)
```

{

putpixel(x,y,1);

x=x+dx;

y=y+dy;

}

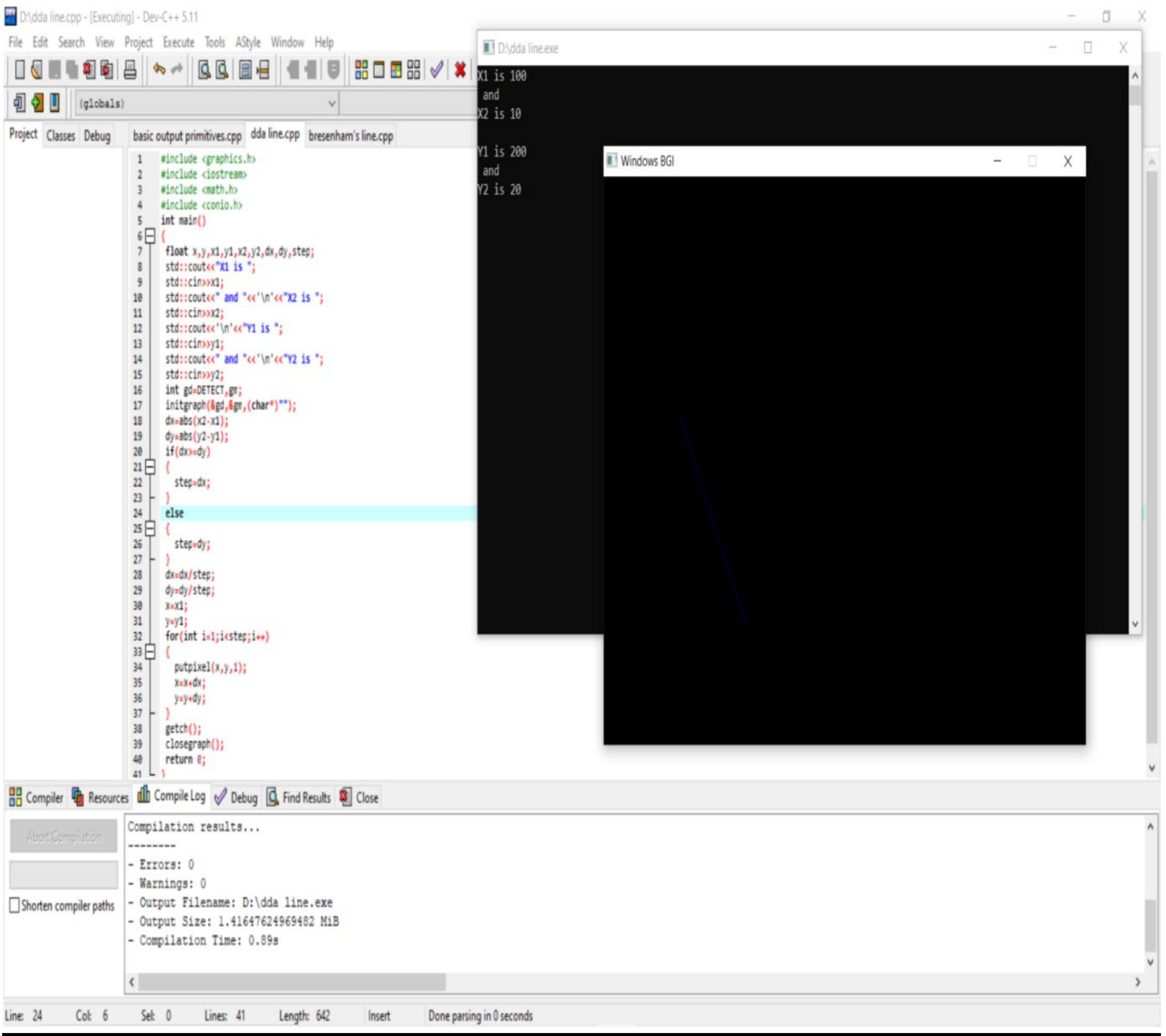
getch();

closegraph();

return 0;

}

<u>OUTPUT:</u>


D:\dda line.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
basic output primitives.cpp
dda line.cpp
bresenham's line.cpp
D:\dda line.exe
Windows BGI
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 24 Col: 6 Sel: 0 Lines: 41 Length: 642 Insert Done parsing in 0 seconds

```
#include <graphics.h>
#include <iostream>
#include <math.h>
#include <conio.h>
int main()
{
 float x,y,x1,y1,x2,y2,dx,dy,step;
 std::cout<<"X1 is ";
 std::cin>>x1;
 std::cout<<" and "<<'\n'<<"X2 is ";
 std::cin>>x2;
 std::cout<<'\n'<<"Y1 is ";
 std::cin>>y1;
 std::cout<<" and "<<'\n'<<"Y2 is ";
 std::cin>>y2;
 int gd=DETECT,gm;
 initgraph(&gd,&gm,(char*)"");
 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;
 for(int i=1;i<step;i++)
 {
   putpixel(x,y,1);
   x=x+dx;
   y=y+dy;
 }
 getch();
 closegraph();
 return 0;
}
```

```
X1 is 100
 and
X2 is 10

Y1 is 200
 and
Y2 is 20
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\dda line.exe
- Output Size: 1.41647624969482 MiB
- Compilation Time: 0.89s
```

LAB NO .3

Question :- Write a program to imlement Bresenham's line drawing algorithm.

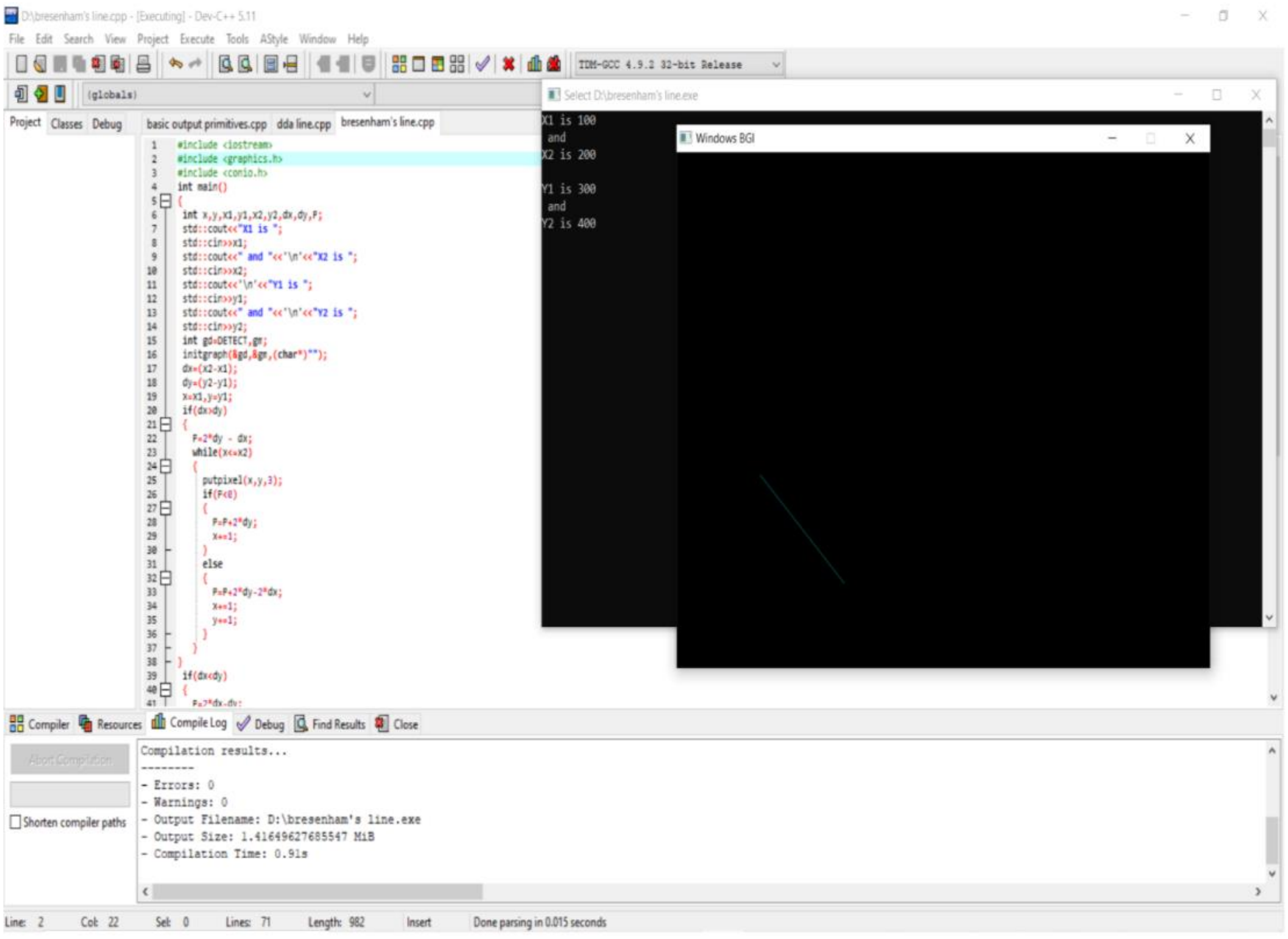
Code:

```
#include <iostream>
#include <graphics.h>
#include <conio.h>
int main()
{
int x,y,x1,y1,x2,y2,dx,dy,P;
std::cout<<"X1 is ";
std::cin>>x1;
std::cout<<" and "<<'\n'<<"X2 is ";
std::cin>>x2;
std::cout<<'\n'<<"Y1 is ";
std::cin>>y1;
std::cout<<" and "<<'\n'<<"Y2 is ";
std::cin>>y2;
int gd=DETECT,gm;
initgraph(&gd,&gm,(char*)"");
dx=(x2-x1);
dy=(y2-y1);
x=x1,y=y1;
if(dx>dy)

{
P=2*dy - dx;
while(x<=x2)
{
putpixel(x,y,3);
if(P<0)
{
```

```
P=P+2*dy;
x+=1;
}
else
{
P=P+2*dy-2*dx;
x+=1;
y+=1;
}
}
}
if(dx<dy)
{
P=2*dx-dy;
while(y<=y2)
{
putpixel(x,y,3);
if(P<0)
{
P=P+2*dx;
y+=1;
}
else
{
P=P+2*dx-2*dy;
x+=1;
y+=1;
}

}
}
if(dx==dy)
```

```
{
while(x<=x2)
{
putpixel(x,y,3);
x+=1;
y+=1;
}
}


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

OUTPUT:



D:\bresenham's line.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 32-bit Release
(globals)
Project Classes Debug
basic output primitives.cpp dda line.cpp bresenham's line.cpp
Select D:\bresenham's line.exe
Windows BGI
Compiler Resources Compile Log Debug Find Results Close
Abort Compilation
Shorten compiler paths
Line: 2 Col: 22 Sel: 0 Lines: 71 Length: 982 Insert Done parsing in 0.015 seconds

```
#include <iostream>
#include <graphics.h>
#include <conio.h>
int main()
{
 int x,y,x1,y1,x2,y2,dx,dy,P;
 std::cout<<"X1 is ";
 std::cin>>x1;
 std::cout<<" and "<<'\n'<<"X2 is ";
 std::cin>>x2;
 std::cout<<'\n'<<"Y1 is ";
 std::cin>>y1;
 std::cout<<" and "<<'\n'<<"Y2 is ";
 std::cin>>y2;
 int gd=DETECT,gm;
 initgraph(&gd,&gm,(char*)"");
 dx=(x2-x1);
 dy=(y2-y1);
 x=x1,y=y1;
 if(dx>dy)
 {
   P=2*dy - dx;
   while(x<=x2)
   {
     putpixel(x,y,3);
     if(P<0)
     {
       P=P+2*dy;
       x+=1;
     }
     else
     {
       P=P+2*dy-2*dx;
       x+=1;
       y+=1;
     }
   }
 }
 if(dx<dy)
 {
   P=2*dx-dy;
```

```
X1 is 100
 and
X2 is 200

Y1 is 300
 and
Y2 is 400
```

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: D:\bresenham's line.exe
- Output Size: 1.41649627685547 MiB
- Compilation Time: 0.91s
```

LAB ASSIGNMENT NO. 04

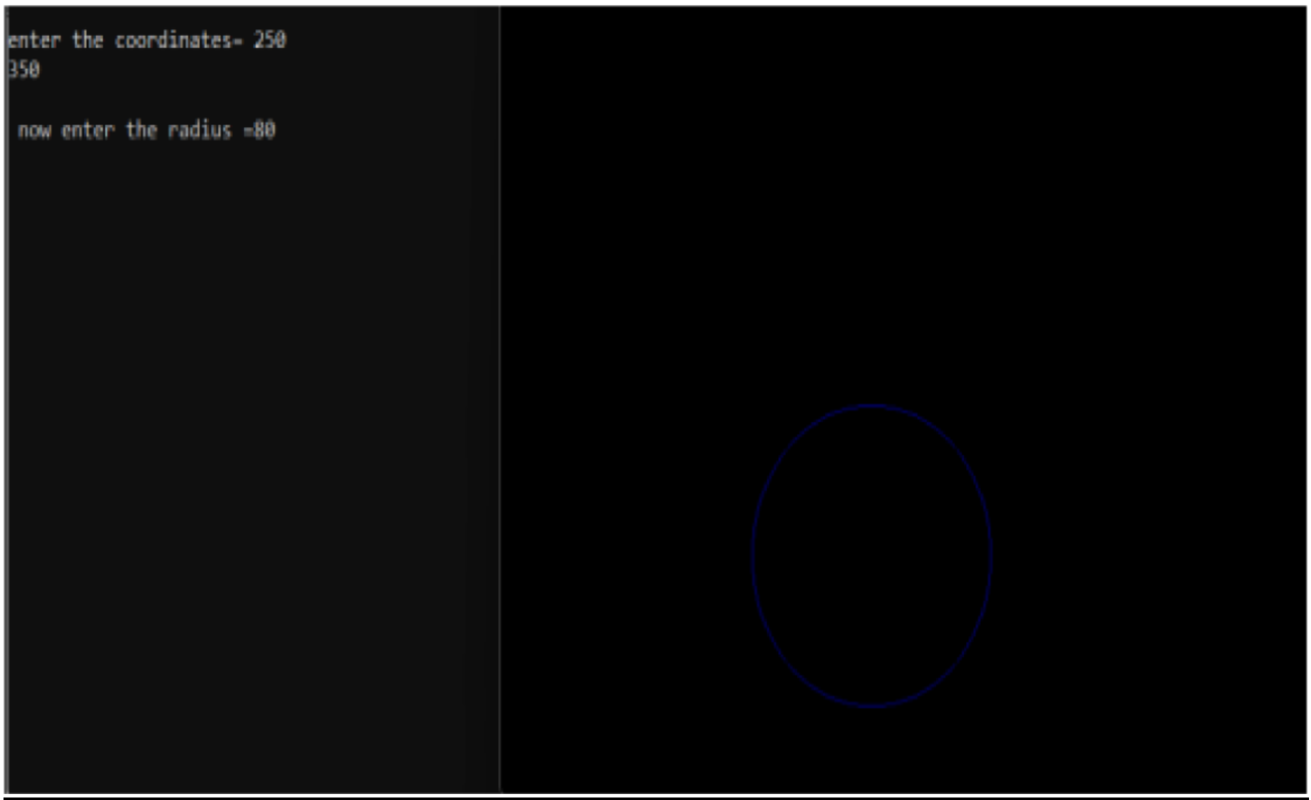
QUESTION:- WAP to implement Mid Point Circle Drawing Algorithm.

CODE:

```
#include<conio.h>
#include<graphics.h>
#include<iostream>
int main()
{
int x,y,x_mid,y_mid,r,d;
int g_mode,g_driver=DETECT;
initgraph(&g_driver,&g_mode,(char*)””);
std::cout<<”enter the coordinates=”;
std::cin>>x_mid>>y_mid;
std::cout<<”\n now enter the radius=”;
std::cin>>r;
x=0;
y=r;
d=l-r;
do
{
putpixel(x_mid+x,y_mid+y,1);
putpixel(x_mid+y,y_mid+x,1);
putpixel(x_mid-y,y_mid+x,1);
putpixel(x_mid-x,y_mid+y,1);
putpixel(x_mid-x,y_mid-y,1);
putpixel(x_mid-y,y_mid-x,1);
putpixel(x_mid+y,y_mid-x,1);
putpixel(x_mid+x,y_mid-y,1);
if(d<0) {
```

```
d+=(2*x)+1;
}
else{
y=y-1;
d+=(2*x)-(2*y)+1;
}
x=x+1;
}while(y>x);
getch();
}
```

<u>OUTPUT:</u>



```
enter the coordinates= 250
350

 now enter the radius =80
```

LAB ASSIGNMENT NO. 05

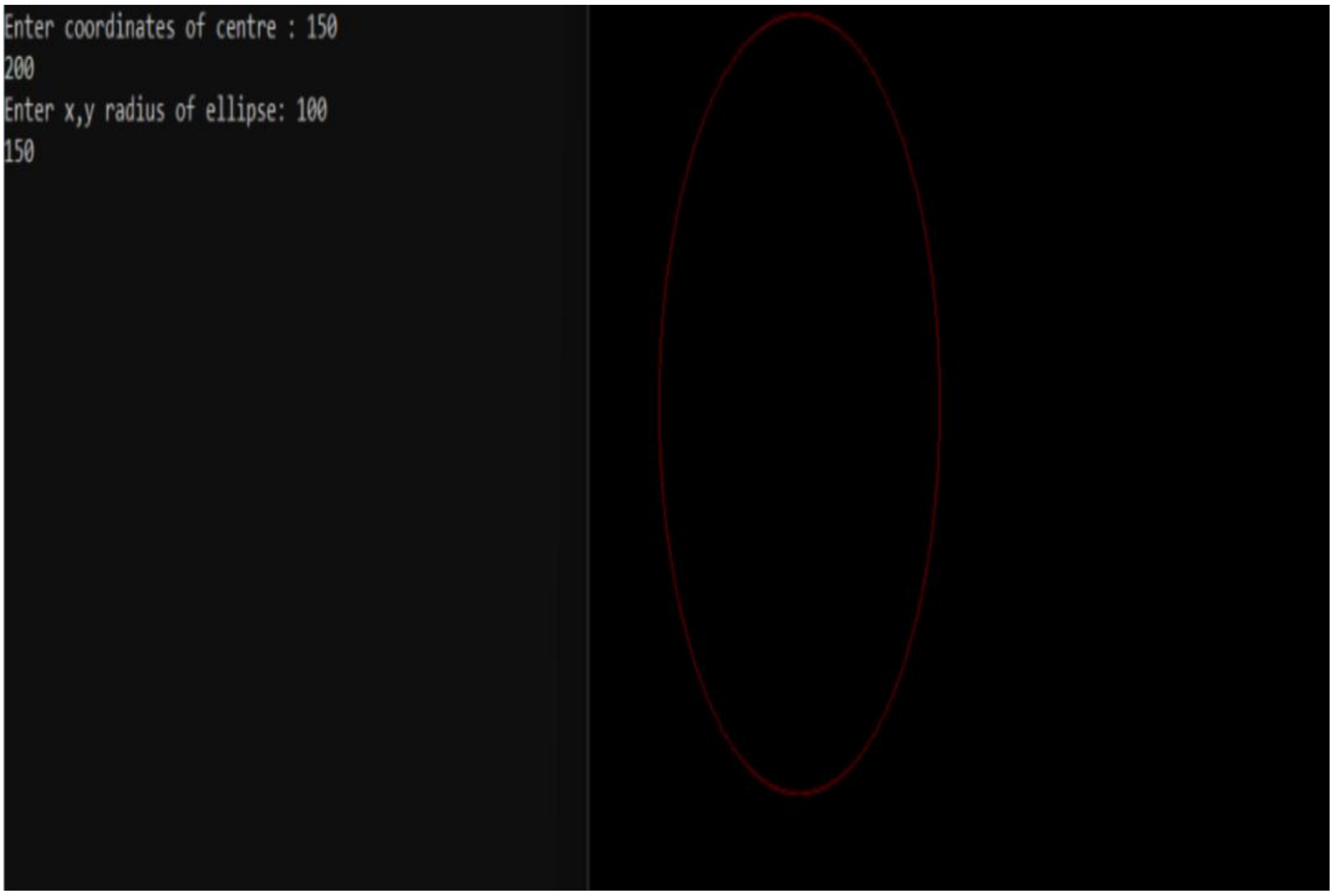
QUESTION:- **WAP to implement Mid Point Ellipse drawing Algorithm.**

CODE:

```
#include<graphics.h>
#include<stdlib.h>
#include<iostream>
#include<conio.h>
int main()
{
int gd = DETECT, gm;
int xc,yc,x,y;float p;
long rx,ry;
initgraph(&gd, &gm, (char*)"");
std::cout<<"\nEnter coordinates of centre : ";
std::cin>>xc>>yc;
std::cout<<"Enter x,y radius of ellipse: ";
std::cin>>rx>>ry;
//Region 1
p=ry*ry-rx*rx*ry+rx*rx/4;
x=0;y=ry;
while(2.0*ry*ry*x <= 2.0*rx*rx*y)
{
if(p < 0)
{
x++;
p = p+2*ry*ry*x+ry*ry;
}
else
{
```

```
x++;y--;
p = p+2*ry*ry*x-2*rx*rx*y-ry*ry;
}
putpixel(xc+x,yc+y,RED);
putpixel(xc+x,yc-y,RED);
putpixel(xc-x,yc+y,RED);
putpixel(xc-x,yc-y,RED);
}
//Region 2
p=ry*ry*(x+0.5)*(x+0.5)+rx*rx*(y-1)*(y-1)-rx*rx*ry*ry;

while(y > 0)
{
if(p <= 0)
{
x++;y--;
p = p+2*ry*ry*x-2*rx*rx*y+rx*rx;
}
else
{
y--;
p = p-2*rx*rx*y+rx*rx;
}
putpixel(xc+x,yc+y,RED);
putpixel(xc+x,yc-y,RED);
putpixel(xc-x,yc+y,RED);
putpixel(xc-x,yc-y,RED);
}
getch();
closegraph();
}
```

OUTPUT:



```
Enter coordinates of centre : 150
200
Enter x,y radius of ellipse: 100
150
```

LAB ASSIGNMENT NO. 06

QUESTION:- WAP to implement the Boundary fill algorithm.

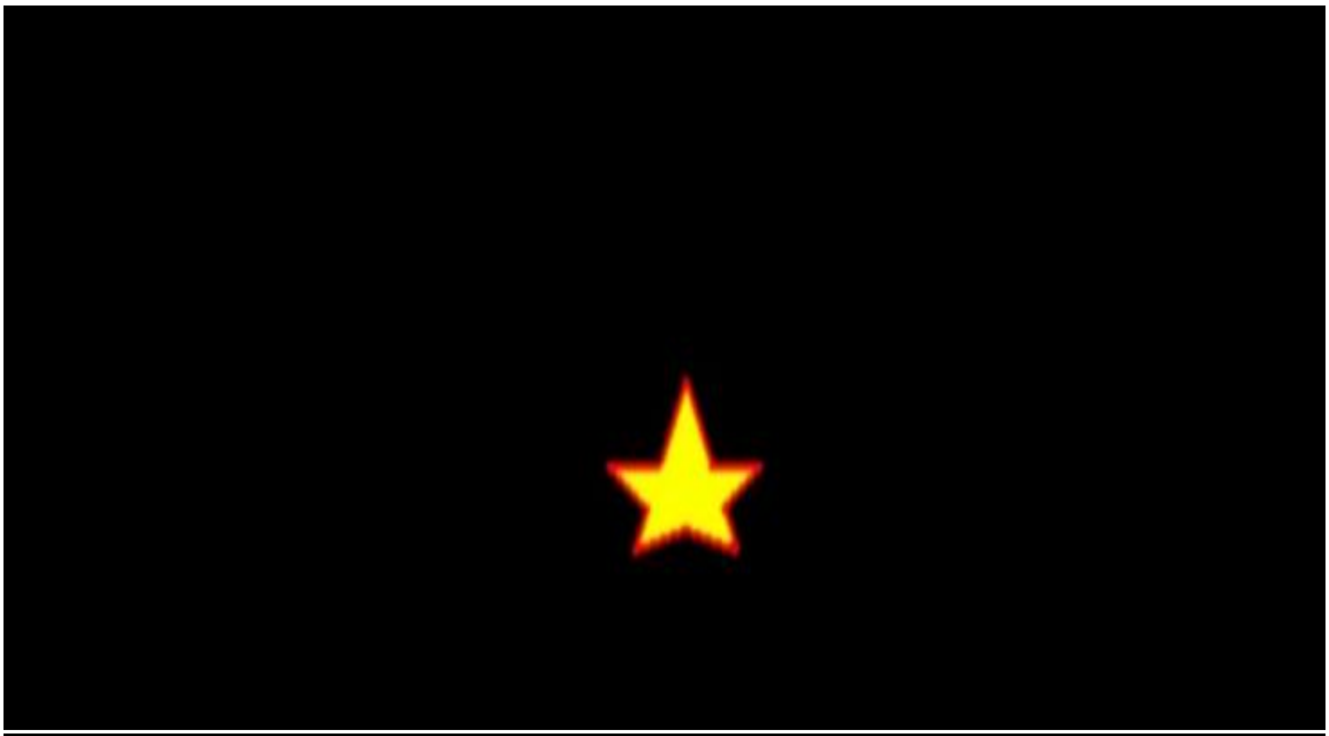
CODE:

```
import pygame
WHITE = (255, 255, 255, 255)
RED = (255, 0, 0, 255)
BLACK = (0, 0, 0, 255)
YELLOW = (255, 255, 0, 255)
def put_pixel(surface, p, color):
```

```
surface.set_at(p, color)
def boundary_fill(surface, p, fill_color, boundary_color):
'''4-way boundary fill'''
if surface.get_at(p) != boundary_color and surface.get_at(p) != fill_color:
put_pixel(surface, p, fill_color)
dx = [0, 1, 0, -1]
dy = [-1, 0, 1, 0]
for i in range(len(dx)):
boundary_fill(surface, (p[0] + dx[i], p[1] + dy[i]), fill_color, boundary_color)
def main():
pygame.init()
screen = pygame.display.set_mode((600, 200))
pygame.display.set_caption('2K19/CO/394-Sundeep Chand: Boundary Fill Algorithm')
star_coordinates = [
(300, 100),
(290, 120),
(270, 120),
(285, 130),
(280, 140),
(300, 135),
(320, 140),
(315, 130),
(330, 120),
(310, 120),
]
pygame.draw.polygon(screen, RED, star_coordinates, 2)
boundary_fill(screen, (300, 110), YELLOW, RED)
pygame.display.flip()
clock = pygame.time.Clock()
quit = False
while not quit:
for event in pygame.event.get():
```

```
if event.type == pygame.QUIT:
quit = True
clock.tick(60)
if __name__ == '__main__':
main()
```

OUTPUT:



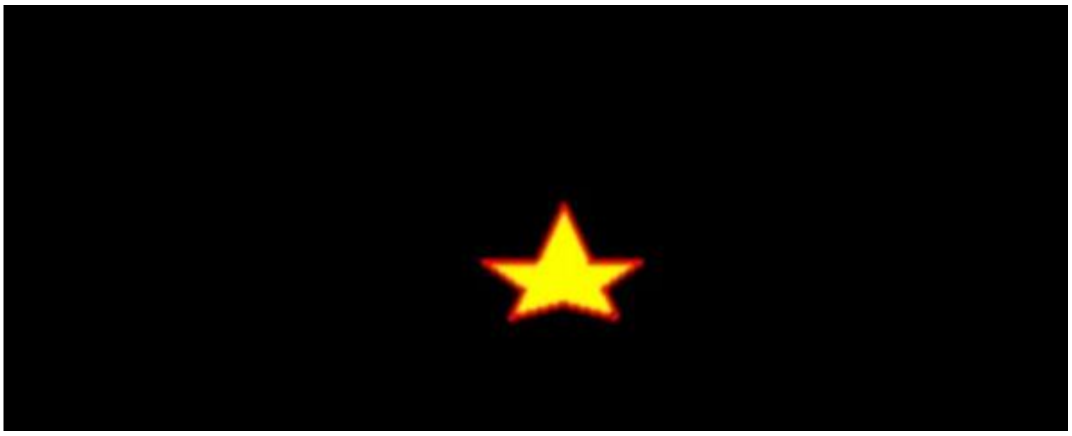
LAB ASSIGNMENT NO. 07

QUESTION:- **WAP to implement the Flood fill algorithm.**

CODE:

```
import pygame
WHITE = (255, 255, 255, 255)
RED = (255, 0, 0, 255)
BLACK = (0, 0, 0, 255)
YELLOW = (255, 255, 0, 255)
SCREEN_WIDTH = 600
SCREEN_HEIGHT = 200
def put_pixel(surface, p, color):
surface.set_at(p, color)
def flood_fill(surface, p, fill_color, prev_color=None):
'''4-way flood fill'''
if p[0] < 0 or p[1] < 0 or p[0] > SCREEN_WIDTH or p[1] > SCREEN_HEIGHT:
return
if prev_color == None:
prev_color = surface.get_at(p)
if surface.get_at(p) == prev_color:
put_pixel(surface, p, fill_color)
dx = [0, 1, 0, -1]
dy = [-1, 0, 1, 0]
for i in range(len(dx)):
flood_fill(surface, (p[0] + dx[i], p[1] + dy[i]), fill_color, prev_color)
def main():
pygame.init()
screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption('2K19/CO/394-Sundeep Chand: Flood Fill Algorithm')
star_coordinates = [
(300, 100),
(290, 120),
(270, 120),
(285, 130),
(280, 140),
(300, 135),
(320, 140),
(315, 130),
(330, 120),
(310, 120),
]
pygame.draw.polygon(screen, RED, star_coordinates, 2)
flood_fill(screen, (300, 110), YELLOW)
# put_pixel(screen, (300, 110), WHITE)
pygame.display.flip()
clock = pygame.time.Clock()
quit = False
while not quit:
for event in pygame.event.get():
if event.type == pygame.QUIT:
quit = True
clock.tick(60)
if __name__ == '__main__':
main()
```

OUTPUT:



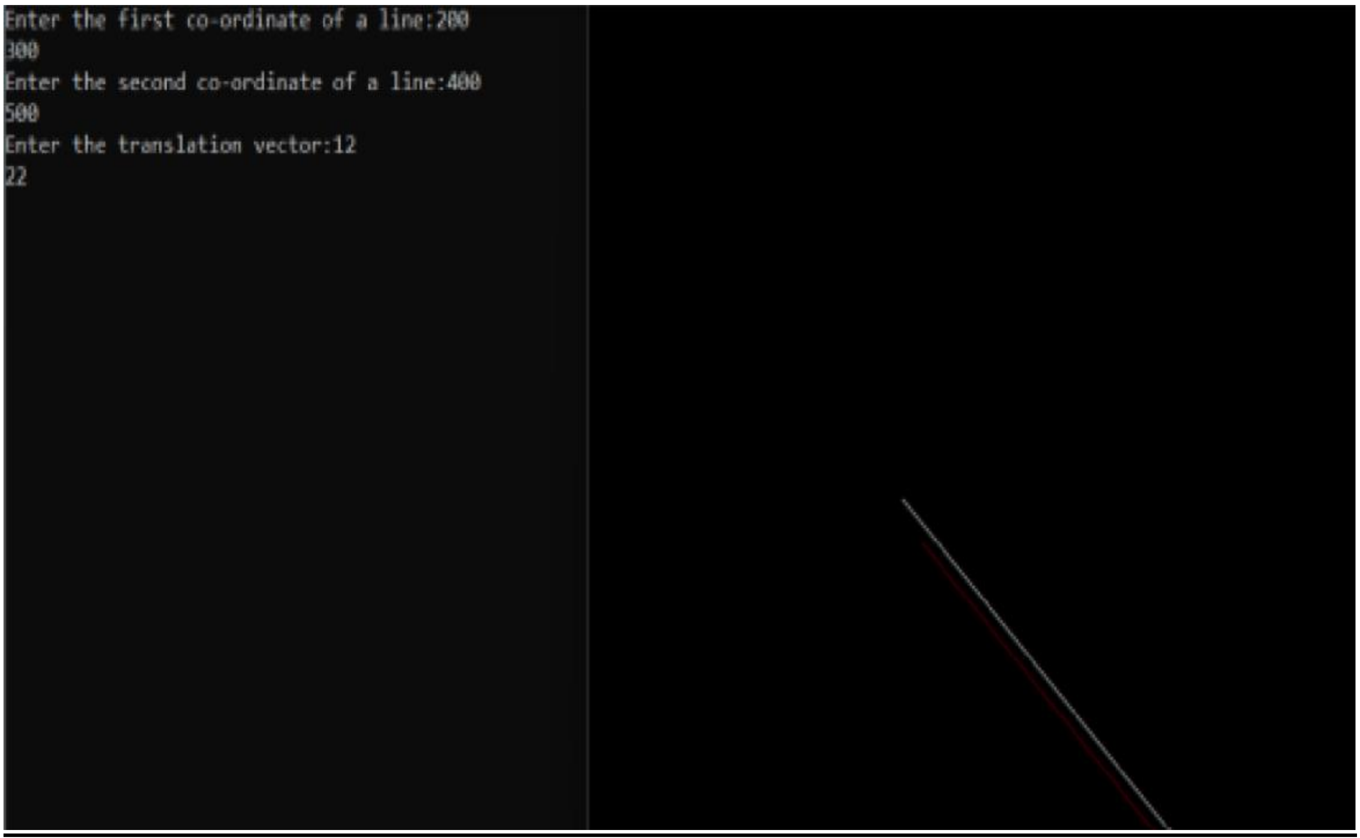
LAB ASSIGNMENT NO. 08

QUESTION:- WAP to implement the Translation an object.

CODE:

```
#include <iostream.h>
#include <conio.h>
#include <graphics.h>
void main()
{
int gd=DETECT,gm,x1,x2,y1,y2,tx,ty;
initgraph(&gd,&gm,(char*)"");
cout<<"Enter the first co-ordinate of a line:";
cin>>x1>>y1;
cout<<"Enter the second co-ordinate of a line:";
cin>>x2>>y2;
line(x1,y1,x2,y2);
cout<<"Enter the translation vector:";
cin>>tx>>ty;
setcolor(RED);
x1=x1+tx;
y1=y1+ty;
x2=x2+tx;
y2=y2+ty;
line(x1,y1,x2,y2);
getch();
closegraph();
}
```

OUTPUT:



```
Enter the first co-ordinate of a line:200
300
Enter the second co-ordinate of a line:400
500
Enter the translation vector:12
22
```

LAB ASSIGNMENT NO. 09

QUESTION:- WAP to implement the Rotation an object.

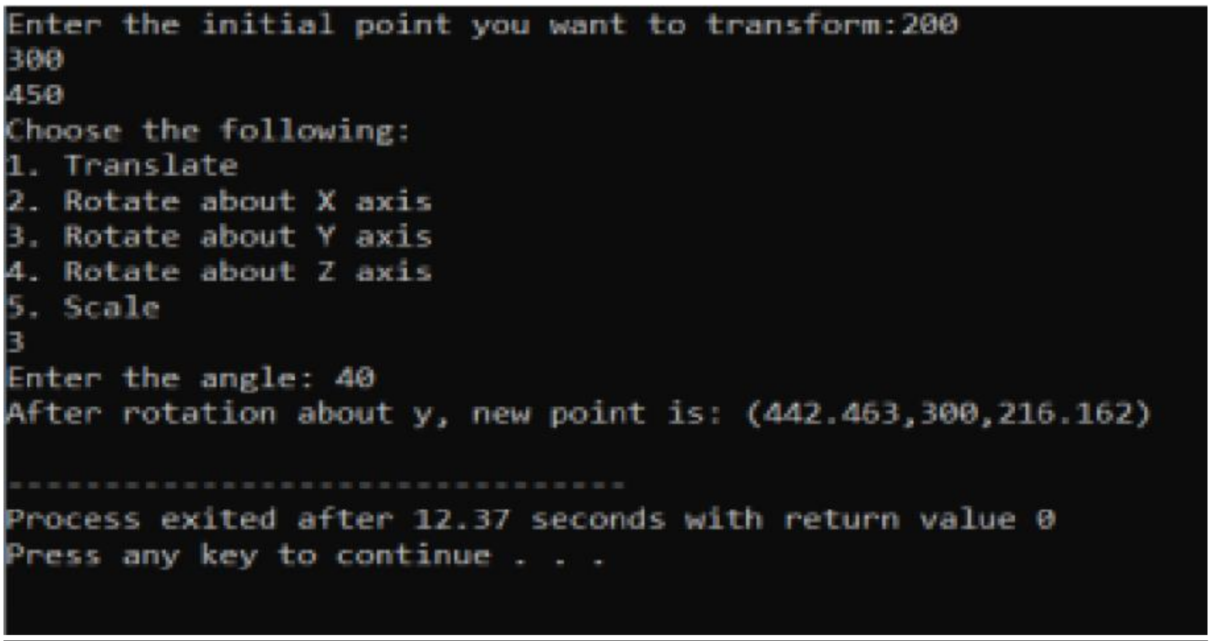
CODE:

```
#include <iostream>
#include <graphics.h>
#include <cmath>
using namespace std;
typedef struct {
float x;
float y;
float z;
```

```
}Point;
Point points;
float temp = 0;
void showPoint(){
cout<<"("<<points.x<<","<<points.y<<","<<points.z<<")"<<endl;
}
void translate(float tx, float ty, float tz){
points.x += tx;
points.y += ty;
points.z += tz;

cout<<"After Translation, new point is :";
showPoint();
}
void rotatex(float angle){
angle = angle * M_PI / 180.0;
temp = points.y;
points.y = points.y * cos(angle) - points.z * sin(angle);
points.z = temp * sin(angle) + points.z * cos(angle);
cout<<"After rotation about x, new point is: ";
showPoint();
}
void rotatey(float angle){
angle = (angle * M_PI) / 180.0;
temp = points.z;
points.z = points.z * cos(angle) - points.x * sin(angle);
points.x = temp * sin(angle) + points.x * cos(angle);
cout<<"After rotation about y, new point is: ";
showPoint();
}
void rotatez(float angle){
angle = angle * M_PI / 180.0;
```

```
temp = points.x;
points.x = points.x * cos(angle) - points.y * sin(angle);
points.y = temp * sin(angle) + points.y *cos(angle);
cout<<"After rotation about z, new point is: ";
showPoint();
}
void scale(float sf, float xf, float yf, float zf){
points.x = points.x * sf + (1 - sf) * xf;
points.y = points.y * sf + (1 - sf) * yf;
points.z = points.z * sf + (1 - sf) * zf;
cout<<"After scaling, new point is: ";
showPoint();
}

int main()
{
float tx = 0, ty = 0, tz = 0;
float sf = 0, xf = 0, yf = 0, zf = 0;
int choose;
float angle;
cout<<"\n\nEnter the initial point you want to transform:";
cin>>points.x>>points.y>>points.z;
cout<<"Choose the following: "<<endl;
cout<<"1. Translate"<<endl;
cout<<"2. Rotate about X axis"<<endl;
cout<<"3. Rotate about Y axis"<<endl;
cout<<"4. Rotate about Z axis"<<endl;
cout<<"5. Scale"<<endl;
cin>>choose;
switch(choose){
case 1:
cout<<"Enter the value of tx, ty and tz: ";
cin>>tx>>ty>>tz;
translate(tx, ty, tz);
break;
case 2:
cout<<"Enter the angle: ";
cin>>angle;
rotatex(angle);
break;
case 3:
cout<<"Enter the angle: ";
cin>>angle;
rotatey(angle);
break;
case 4:
cout<<"Enter the angle: ";
cin>>angle;
rotatez(angle);
break;
case 5:
cout<<"Enter the value of sf, xf, yf and zf: ";
cin>>sf>>xf>>yf>>zf;
scale(sf, xf, yf, zf);
break;
default:
break;
}
return 0;
}
```

OUTPUT:



```
Enter the initial point you want to transform:200
300
450
Choose the following:
1. Translate
2. Rotate about X axis
3. Rotate about Y axis
4. Rotate about Z axis
5. Scale
3
Enter the angle: 40
After rotation about y, new point is: (442.463,300,216.162)

--------------------------------
Process exited after 12.37 seconds with return value 0
Press any key to continue . . .
```

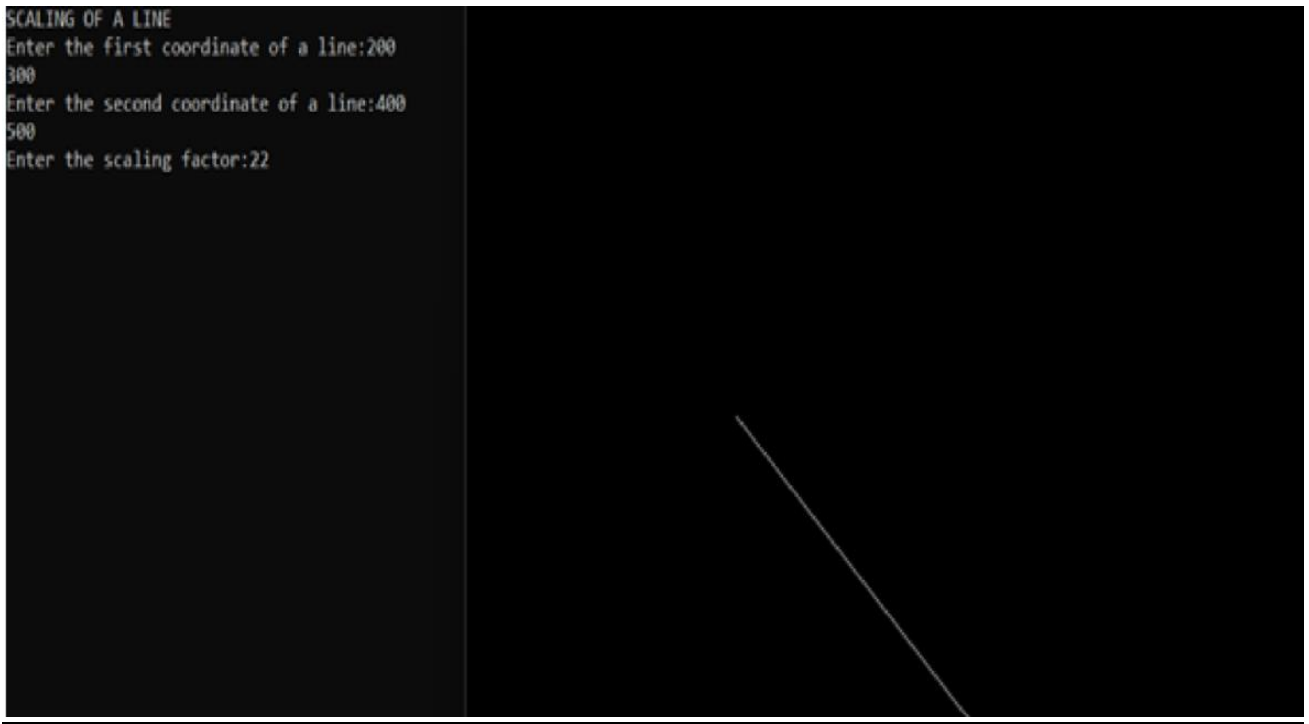
LAB ASSIGNMENT NO. 10

QUESTION:- **WAP to implement the Scaling an object.**

CODE:

```
#include <iostream>
#include <conio.h>
#include <graphics.h>
int main()
{
int gd=DETECT,gm;
float x1,y1,x2,y2,sx,sy;
initgraph(&gd,&gm,(char*)"");
std::cout<<"name:sristi mitra\n\n";
std::cout<<"roll 2k19.co.389\n\n";
std::cout<<"SCALING OF A LINE\n";
std::cout<<"Enter the first coordinate of a line:";
std::cin>>x1>>y1;

std::cout<<"Enter the second coordinate of a line:";
std::cin>>x2>>y2;
line(x1,y1,x2,y2);
std::cout<<"Enter the scaling factor:";
std::cin>>sx>>sy;
setcolor(RED);
x1=x1*sx;
y1=y1*sy;
x2=x2*sx;
y2=y2*sy;
line(x1,y1,x2,y2);
getch();
closegraph();
}
```

OUTPUT:



```
SCALING OF A LINE
Enter the first coordinate of a line:200
300
Enter the second coordinate of a line:400
500
Enter the scaling factor:22
```

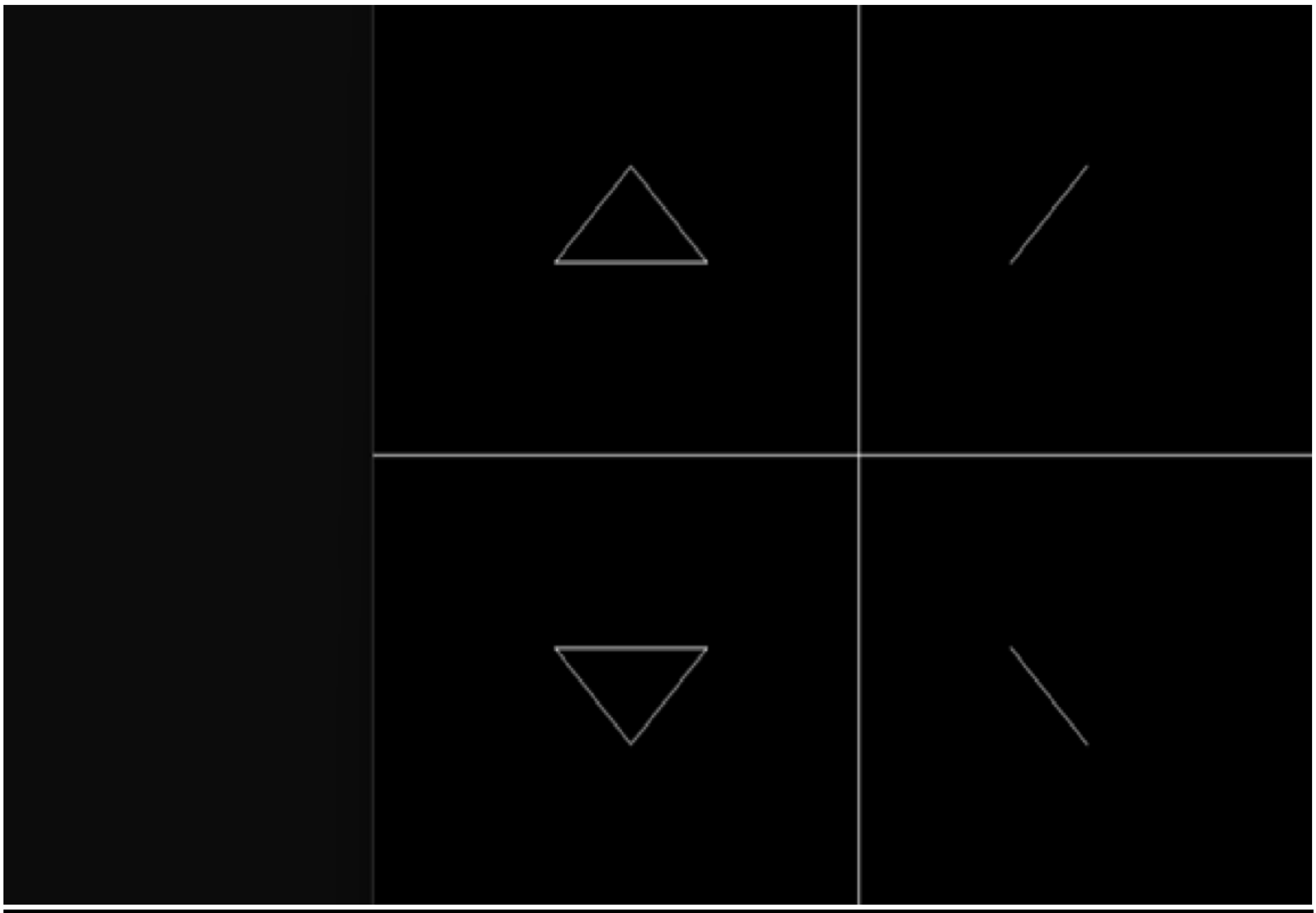
LAB ASSIGNMENT NO. 11

QUESTION:- WAP to implement Reflection an object.

CODE:

```
#include <iostream.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
void main()
{
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\Tc\\BGI");
int a=getmaxx();

int b=getmaxy();
int y=b/2;
int x=a/2;
line(x,0,x,b);
line(0,y,a,y);
int x1=x+100;
int y1=y-100;
int x2=x+150;
int y2=y-150;
line(x2,y2,x1,y1);
line(x2,y+150,x1,y+100);
line(x-100,y+100,x-200,y+100);
line(x-150,y+150,x-200,y+100);
line(x-100,y+100,x-150,y+150);
line(x-100,y-100,x-150,y-150);
line(x-150,y-150,x-200,y-100);
line(x-200,y-100,x-100,y-100);
getch();
closegraph();
}
```

OUTPUT:



LAB ASSIGNMENT NO. 12

QUESTION:- Write a program to implement Cohen Sutherland Line Clipping Algorithm.

CODE:

```
#include <bits/stdc++.h>
#include <graphics.h>
using namespace std;
int xmin, xmax, ymin, ymax;
struct lines {
int x1, y1, x2, y2;
};
int sign(int x)
{
if (x > 0)
return 1;
else
return 0;
}
void clip(struct lines mylines)
{
int bits[4], bite[4], i, var;
setcolor(RED);
bits[0] = sign(xmin - mylines.x1);
bite[0] = sign(xmin - mylines.x2);
bits[1] = sign(mylines.x1 - xmax);
bite[1] = sign(mylines.x2 - xmax);
bits[2] = sign(ymin - mylines.y1);
bite[2] = sign(ymin - mylines.y2);
bits[3] = sign(mylines.y1 - ymax);
bite[3] = sign(mylines.y2 - ymax);
```

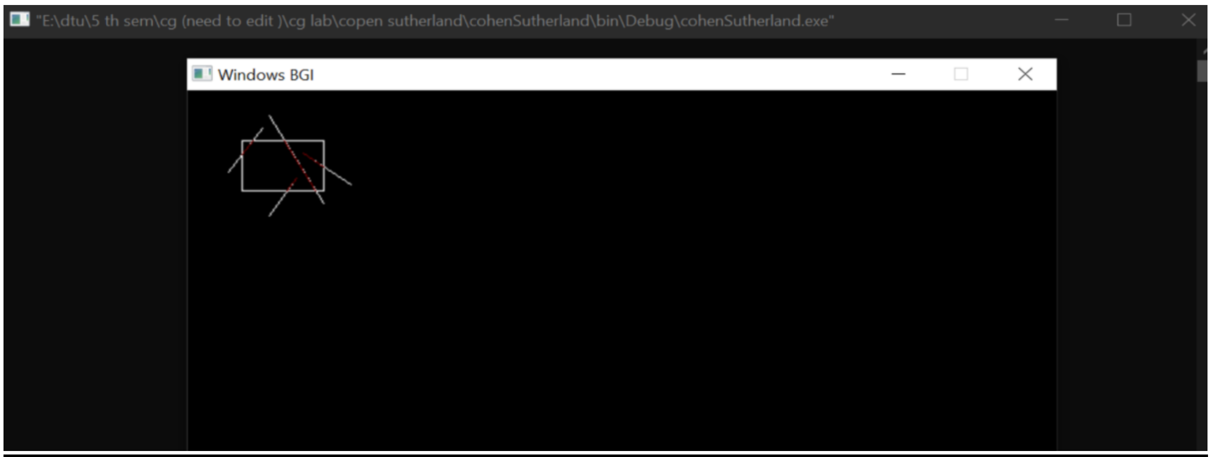
```
string initial = "", end = "", temp = "";
for (i = 0; i < 4; i++) {
if (bits[i] == 0)
initial += '0';
else

initial += '1';
}
for (i = 0; i < 4; i++) {
if (bite[i] == 0)
end += '0';
else
end += '1';
}
float m = (mylines.y2 - mylines.y1) / (float)(mylines.x2 - mylines.x1);
float c = mylines.y1 - m * mylines.x1;

if (initial == end && end == "0000") {
line(mylines.x1, mylines.y1, mylines.x2, mylines.y2);
return;
}

else {
for (i = 0; i < 4; i++) {
int val = (bits[i] & bite[i]);
if (val == 0)
temp += '0';
else
temp += '1';
}
if (temp != "0000")
return;
```

```
for (i = 0; i < 4; i++) {
if (bits[i] == bite[i])
continue;
if (i == 0 && bits[i] == 1) {
var = round(m * xmin + c);
mylines.y1 = var;
mylines.x1 = xmin;
}
if (i == 0 && bite[i] == 1) {
var = round(m * xmin + c);
mylines.y2 = var;
mylines.x2 = xmin;
}
if (i == 1 && bits[i] == 1) {
var = round(m * xmax + c);
mylines.y1 = var;
mylines.x1 = xmax;

}
if (i == 1 && bite[i] == 1) {
var = round(m * xmax + c);
mylines.y2 = var;
mylines.x2 = xmax;
}
if (i == 2 && bits[i] == 1) {
var = round((float)(ymin - c) / m);
mylines.y1 = ymin;
mylines.x1 = var;
}
if (i == 2 && bite[i] == 1) {
var = round((float)(ymin - c) / m);
mylines.y2 = ymin;
mylines.x2 = var;
}
if (i == 3 && bits[i] == 1) {
var = round((float)(ymax - c) / m);
mylines.y1 = ymax;
mylines.x1 = var;
}
if (i == 3 && bite[i] == 1) {
var = round((float)(ymax - c) / m);
mylines.y2 = ymax;
mylines.x2 = var;
}
bits[0] = sign(xmin - mylines.x1);
bite[0] = sign(xmin - mylines.x2);
bits[1] = sign(mylines.x1 - xmax);
bite[1] = sign(mylines.x2 - xmax);
bits[2] = sign(ymin - mylines.y1);
bite[2] = sign(ymin - mylines.y2);
bits[3] = sign(mylines.y1 - ymax);
bite[3] = sign(mylines.y2 - ymax);
}
initial = "", end = "";
for (i = 0; i < 4; i++) {
if (bits[i] == 0)
initial += '0';
else
initial += '1';
}
for (i = 0; i < 4; i++) {
if (bite[i] == 0)
end += '0';
else
```

```
end += '1';
}

if (initial == end && end == "0000") {
line(mylines.x1, mylines.y1, mylines.x2, mylines.y2);
return;
}
else
return;
}
}
int main()
{
int gd = DETECT, gm;
xmin = 40;
xmax = 100;
ymin = 40;
ymax = 80;
initgraph(&gd, &gm, NULL);
line(xmin, ymin, xmax, ymin);
line(xmax, ymin, xmax, ymax);
line(xmax, ymax, xmin, ymax);
line(xmin, ymax, xmin, ymin);
struct lines mylines[4];
mylines[0].x1 = 30;
mylines[0].y1 = 65;
mylines[0].x2 = 55;
mylines[0].y2 = 30;
mylines[1].x1 = 60;
mylines[1].y1 = 20;
mylines[1].x2 = 100;
mylines[1].y2 = 90;
```

```
mylines[2].x1 = 60;
mylines[2].y1 = 100;
mylines[2].x2 = 80;
mylines[2].y2 = 70;
mylines[3].x1 = 85;
mylines[3].y1 = 50;
mylines[3].x2 = 120;
mylines[3].y2 = 75;
for (int i = 0; i < 4; i++) {
line(mylines[i].x1, mylines[i].y1,
mylines[i].x2, mylines[i].y2);
delay(1000);
}
for (int i = 0; i < 4; i++) {
clip(mylines[i]);
delay(1000);
}
delay(4000);
getch();
closegraph();
return 0;
}
```

OUTPUT:


"E:\dtu\5 th sem\cg (need to edit)\cg lab\copen sutherland\cohenSutherland\bin\Debug\cohenSutherland.exe"
Windows BGI