DEPARTMENT OF
COMPUTER SCIENCE & ENGINEERING
Discover. Learn. Empower.

UNIVERSITY INSTITUTE OF ENGINEERING

Department of Computer Science & Engineering

(BE-CSE/IT-6th Sem)

Subject Name: Foundation of Cloud IOT Edge ML

Subject Code: 22CSP-367

Submitted to:

Submitted by:

Name: Vinod Kumar Saini

UID: 22BCS14967

Section: 641 / B

Experiment 5

Student Name: Vinod Kumar Saini **UID:** 22BCS14967

Branch: CSE **Section/Group:** 641/B

Semester: 6th **Date of Performance:** 20/02/25

Subject Code: 22CSH-367

Subject Name: Foundation of Cloud IOT Edge ML lab

1. **Aim** : Set up a system using IoT sensor data to AWS IoT Core and store it in an S3 bucket.

2. **Objective :** To demonstrate the process of integrating IoT sensors with AWS IoT Core, transmitting sensor data, and storing the data in AWS S3 for further analysis.

3. **Software :** AWS

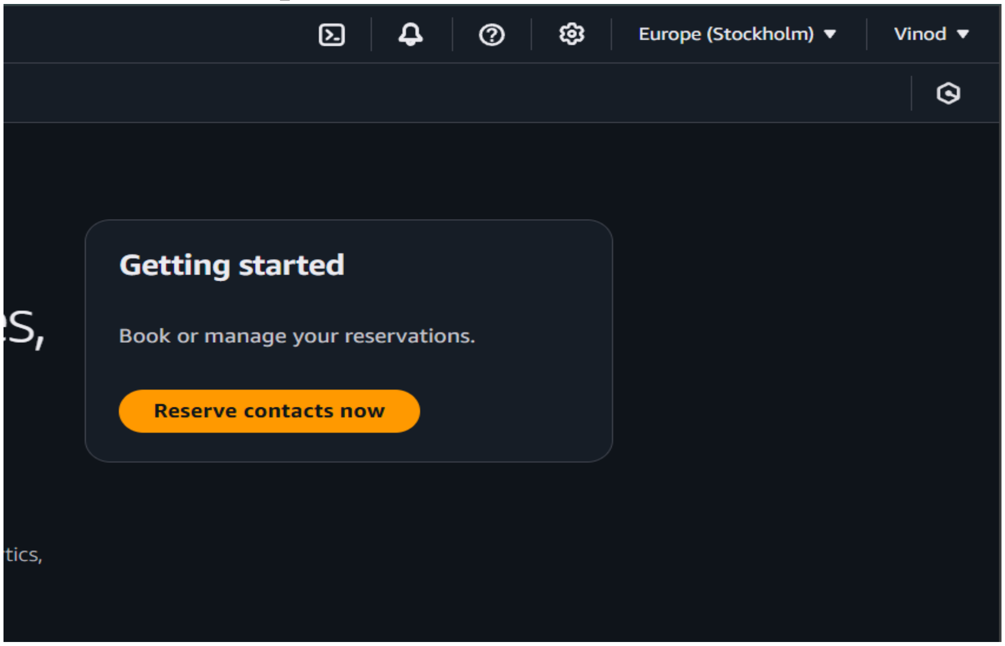
4. **Procedure :**

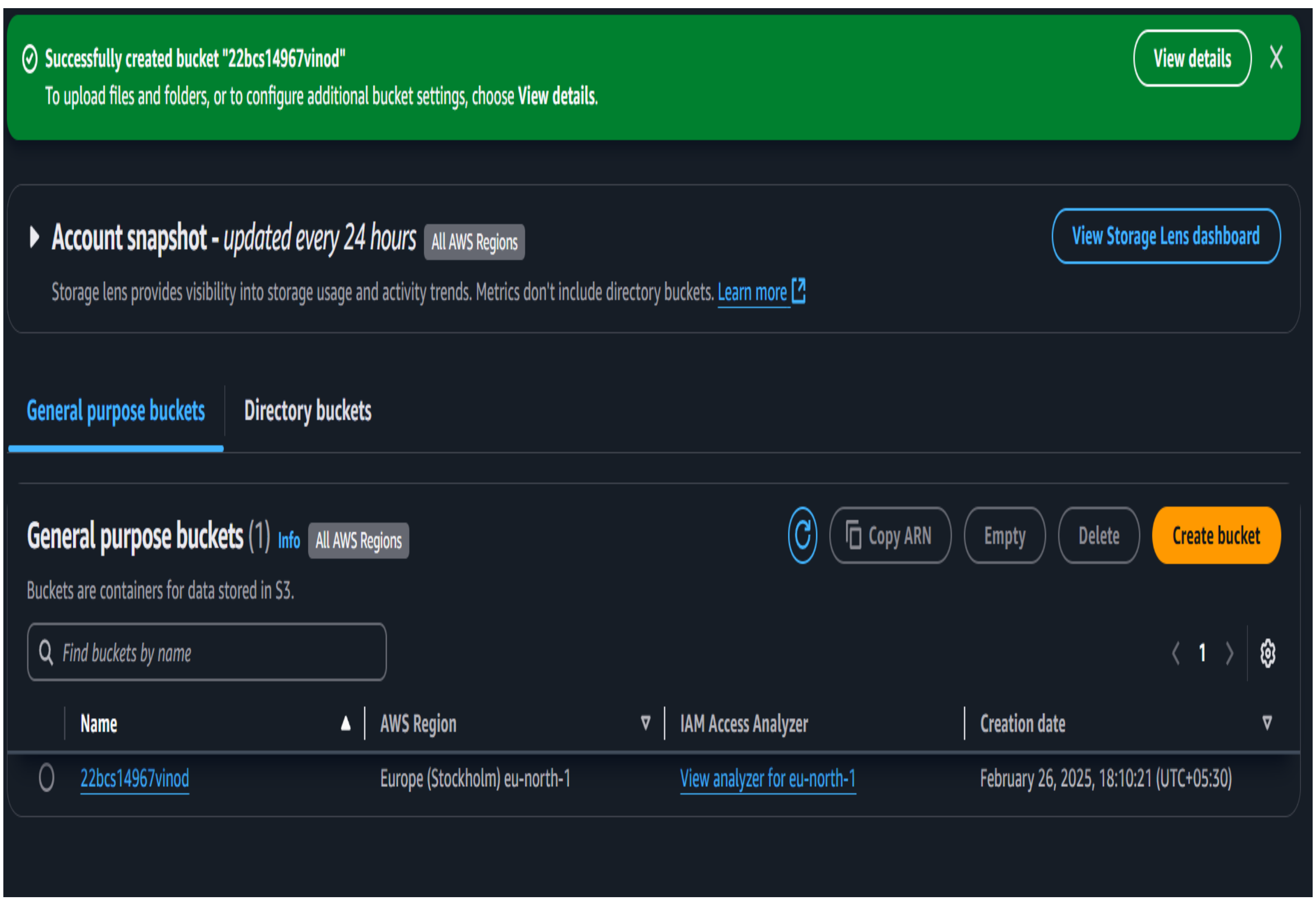
   1. Create S3 Bucket.
      - a Log in to your AWS account, type S3 into the search box in the main menu, and then select S3 from the services menu.
      - b In the left blade of the AWS S3 console, select the Buckets option and then click Create bucket option.

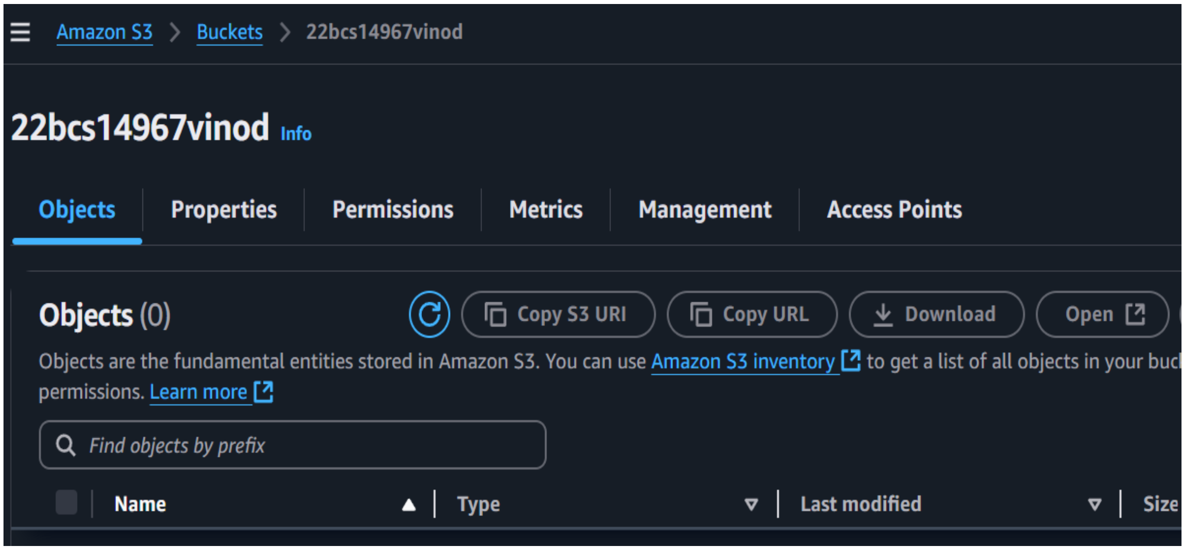
   2. Leave the rest of the options alone and click Create bucket at the bottom.
   3. Create an IoT rule to send data to the S3:

      In the search box, type IoT Core, and then pick IoT Core from the services selection.
   4. In the IoT Core console, on the left blade, click Act, and then on the right blade, click rule.
   5. Finally, click Create in the Rules console to create the rule to transfer IoT data to the S3 bucket.
   6. To add an action, click Store a message in an Amazon S3 bucket, select an action blade, and then click Configure action at the bottom page.

5. **Screenshot of Outputs:**







6. Conclusion / Analysis:

1. As a result, AWS IoT Core with the IoT Rule engine will assist in filtering IoT topics and the storage of data in AWS S3.
2. AWS IoT core can receive and send millions of IoT data at a time, and the AWS IoT Rule engine can filter MQTT topics from IoT devices and send them to other AWS Services and a time stamp.
3. For data backup and archive, AWS S3 will be used.

Experiment 5

Student Name: Vinod Kumar Saini **UID:** 22BCS14967

Branch: CSE **Section/Group:** 641/B

Semester: 6th **Date of Performance:** 20/02/25

Subject Code: 22CSH-352

Subject Name: Computer Graphics with Lab

1. **Aim**: Implement clockwise and anticlockwise rotation of a triangle about a specified point and evaluate the results.

2. **Objective :** To perform and visualize clockwise and anticlockwise rotations of a triangle about a specified point.

3. **Algorithms:**

   1. Input the coordinates of the three vertices of the triangle: (x1,y1)(x1, y1)(x1,y1), (x2,y2)(x2, y2)(x2,y2), (x3,y3)(x3, y3)(x3,y3).
   2. Calculate the centroid (center) of the triangle. The formula for the centroid of a triangle with vertices (x1,y1)(x1, y1)(x1,y1), (x2,y2)(x2, y2)(x2,y2), and (x3,y3)(x3, y3)(x3,y3) is:
      xf=(x1+x2+x3)/3,
      yf=(y1+y2+y3)/3.
   3. Convert angle into degree to radians: rad = (angle *π)/180.
   4. Rotate Each Vertex.
      X= (xi−xf)·cos(rad)−(yi−yf)·sin(rad)+xf
      Y= (xi−xf)·sin(rad)+(yi−yf)·cos(rad)+yf

4. **Code :**

   a) To rotate clockwise

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

```
#include<math.h>
void main()
{
clrscr();
int gd=DETECT,gm;
initgraph(&gd,&gm,"");
int x1,y1,x2,y2,x3,y3;
cout<<"Enter (x1,y1),(x2,y2),(x3,y3) for triangle : ";
cin>>x1>>y1>>x2>>y2>>x3>>y3;
int a[]={x1,y1,x2,y2,x3,y3,x1,y1};
drawpoly(4,a);
int xf=(x1+x2+x3)/3;
int yf=(y1+y2+y3)/3;
float ang;
cout<<"Enter the rotation angle :";
cin>>ang;
float rad=ang*3.1428/180;
int X1=(x1-xf)*cos(rad)-(y1-yf)*sin(rad)+xf;
int Y1=(x1-xf)*sin(rad)+(y1-yf)*cos(rad)+yf;
int X2=(x2-xf)*cos(rad)-(y2-yf)*sin(rad)+xf;
int Y2=(x2-xf)*sin(rad)+(y2-yf)*cos(rad)+yf;
int X3=(x3-xf)*cos(rad)-(y3-yf)*sin(rad)+xf;
int Y3=(x3-xf)*sin(rad)+(y3-yf)*cos(rad)+yf;
int b[]={X1,Y1,X2,Y2,X3,Y3,X1,Y1};
drawpoly(4,b);
getch();
}
```

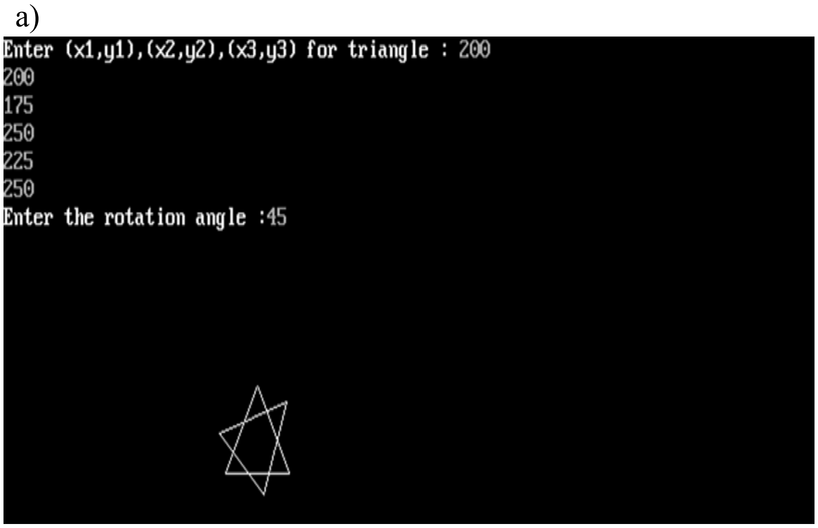
b) To rotate anti-clockwise

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void main()
{
```

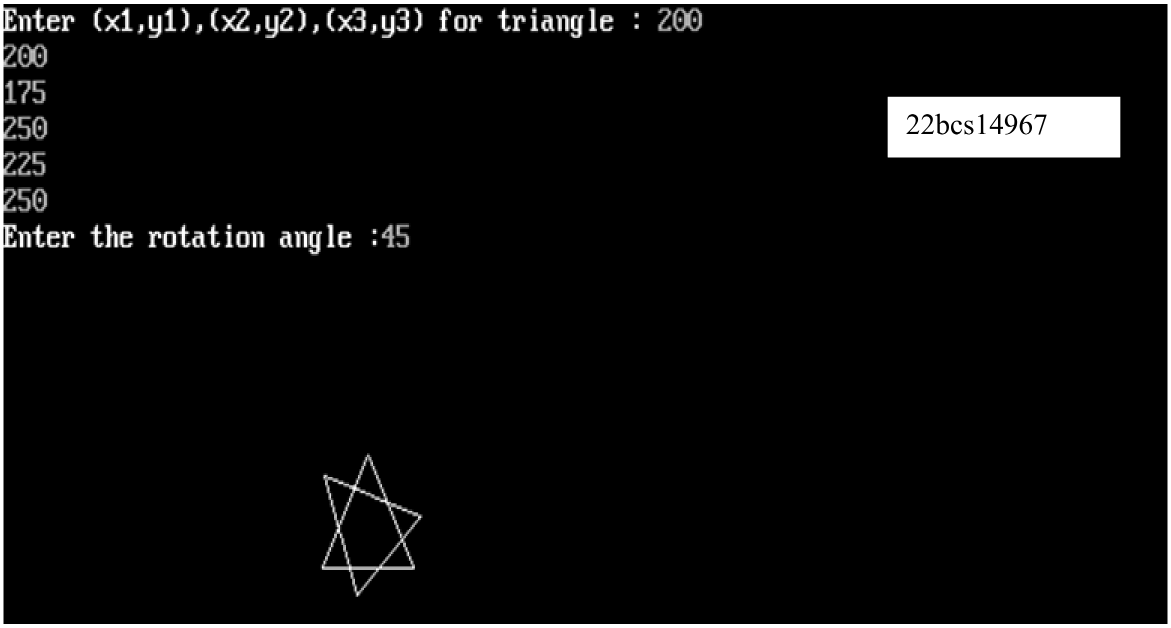
```
clrscr();
int gd=DETECT,gm;
initgraph(&gd,&gm,"");
int x1,y1,x2,y2,x3,y3;
cout<<"Enter (x1,y1),(x2,y2),(x3,y3) for triangle : ";
cin>>x1>>y1>>x2>>y2>>x3>>y3;
int a[]={x1,y1,x2,y2,x3,y3,x1,y1};
drawpoly(4,a);
int xf=(x1+x2+x3)/3;
int yf=(y1+y2+y3)/3;
float ang;
cout<<"Enter the rotation angle :";
cin>>ang;
float rad=ang*3.1428/180;
int X1=(x1-xf)*cos(rad)+(y1-yf)*sin(rad)+xf;
int Y1=-(x1-xf)*sin(rad)+(y1-yf)*cos(rad)+yf;
int X2=(x2-xf)*cos(rad)+(y2-yf)*sin(rad)+xf;
int Y2=-(x2-xf)*sin(rad)+(y2-yf)*cos(rad)+yf;
int X3=(x3-xf)*cos(rad)+(y3-yf)*sin(rad)+xf;
int Y3=-(x3-xf)*sin(rad)+(y3-yf)*cos(rad)+yf;
int b[]={X1,Y1,X2,Y2,X3,Y3,X1,Y1};
drawpoly(4,b);
getch();
}
```

5. **Screenshot of Outputs:**

a)



b)



6. Learning Outcomes:

1. Graphical programming in C++, specifically using the graphics.h library to draw and manipulate shapes (e.g., triangles).
2. The concept of **rotation transformation** in 2D graphics.
3. Understanding Coordinate Transformation in 2D.

Experiment 6

Student Name: Vinod Kumar Saini **UID:** 22BCS14967

Branch: CSE **Section/Group:** 641/B

Semester: 6th **Date of Performance:** 20/02/25

Subject Code: 22CSH-352

Subject Name: Computer Graphics with Lab

1. **Aim**: Analyze and implement the reflection of a point about a line defined by the equation y=mx+c.

2. **Objective :** To implement and analyze the reflection of a point about a straight line defined by the equation y=mx+c.

3. **Algorithms:**

   1. Draw the X-axis and Y-axis at the center of the screen (using line(m/2, 0, m/2, n) for the vertical axis and line(0, n/2, m, n/2) for the horizontal axis).

   2. **Reflect the Line Along the X-axis:**
      Compute the distance between the Y-coordinate of the line's points and the center of the screen along the Y-axis:
      c=n2−y1c = \frac{n}{2} - y1c=2n−y1
      d=n2−y2d = \frac{n}{2} - y2d=2n−y2

      Adjust the Y-coordinates to reflect the points along the X-axis:
      y1=y1+(c×2)y1 = y1 + (c \times 2)y1=y1+(c×2)
      y2=y2+(d×2)y2 = y2 + (d \times 2)y2=y2+(d×2)

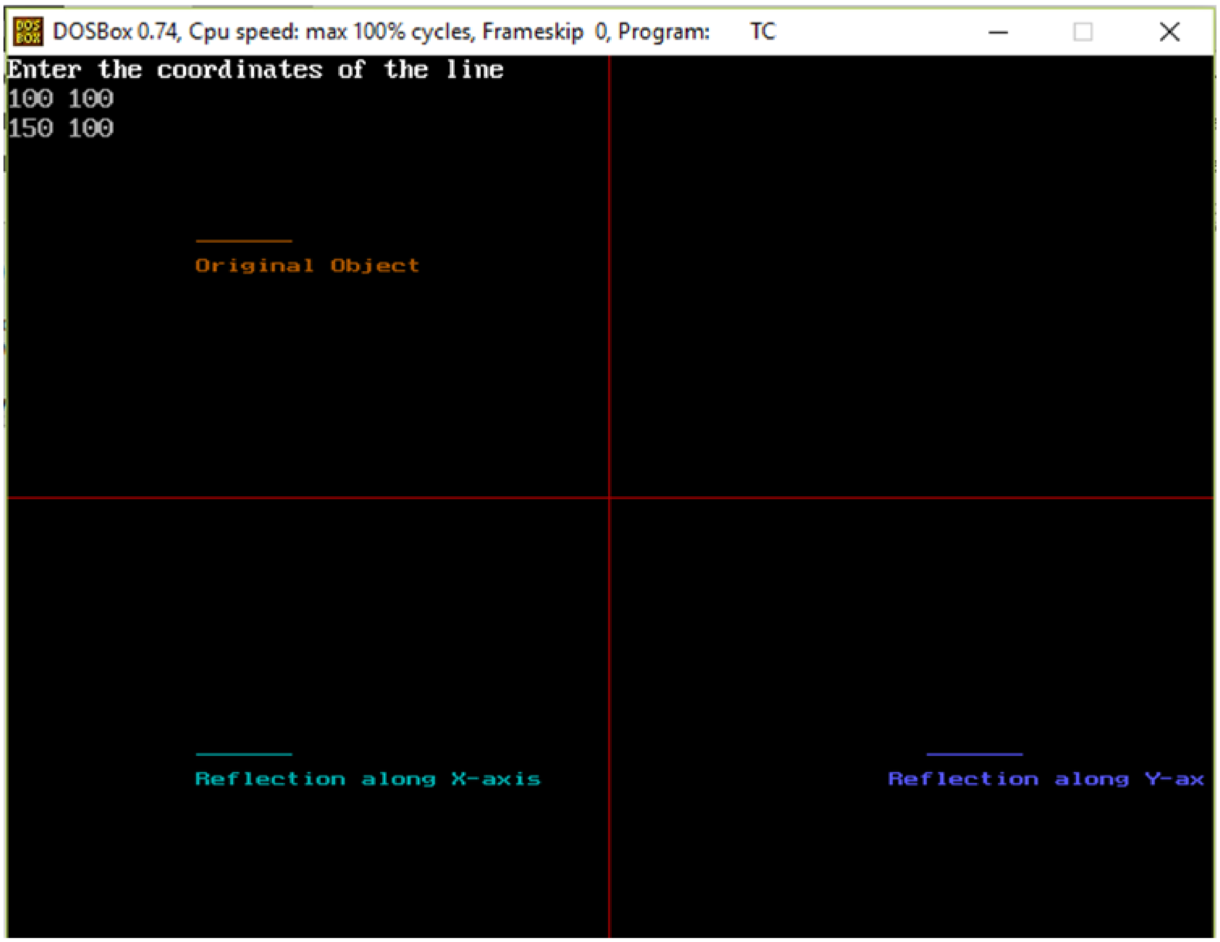
   3. **Reflect the Line Along the Y-axis:**
      Compute the distance between the X-coordinate of the line's points and the center of the screen along the X-axis:
      a=m2−x1a = \frac{m}{2} - x1a=2m−x1
      b=m2−x2b = \frac{m}{2} - x2b=2m−x2

Adjust the X-coordinates to reflect the points along the Y-axis:
x1=x1+(a×2)x1 = x1 + (a \times 2)x1=x1+(a×2)
x2=x2+(b×2)x2 = x2 + (b \times 2)x2=x2+(b×2)

4. Code :

```
#include<iostream.h>
#include<conio.h>
#include<graphics.h>
 void main()
{
        int gd=DETECT,gm;
        initgraph(&gd,&gm,"  ");
        int x1,y1,x2,y2;
        cout<<"Enter the coordinates of the line\n";
        cin>>x1>>y1>>x2>>y2;
        int m=getmaxx();
        int n=getmaxy();
        setcolor(6);
        line(x1,y1,x2,y2);
        outtextxy(x1,y1+10,"Original Object");
        setcolor(4);
        line((m/2),0,(m/2),n);
        line(0,(n/2),m,(n/2));
        setcolor(3);
        int c=(n/2)-y1;
        int d=(n/2)-y2;
        y2=y2+(d*2);
        y1=y1+(c*2);
        line(x2,y2,x1,y1);
        outtextxy(x1,y1+10,"Reflection along X-axis");
        setcolor(9);
        int a=(m/2)-x1;
        int b=(m/2)-x2;
        x1=x1+(a*2);
        x2=x2+(b*2);
        line(x1,y1,x2,y2);
        outtextxy(x2-20,y2+10,"Reflection along Y-axis");
        getch();
        closegraph();
}
```

5. Screenshot of Output



6. Learning Outcomes:

1. Graphical programming in C++, specifically using the graphics.h library to draw and manipulate shape.
2. The concept of **reflection** in 2D graphics.