## 1. WAP to implement DDA algorithm in C for:

1. |m|>1 positive slope
2. |m|<1 negative slope

---

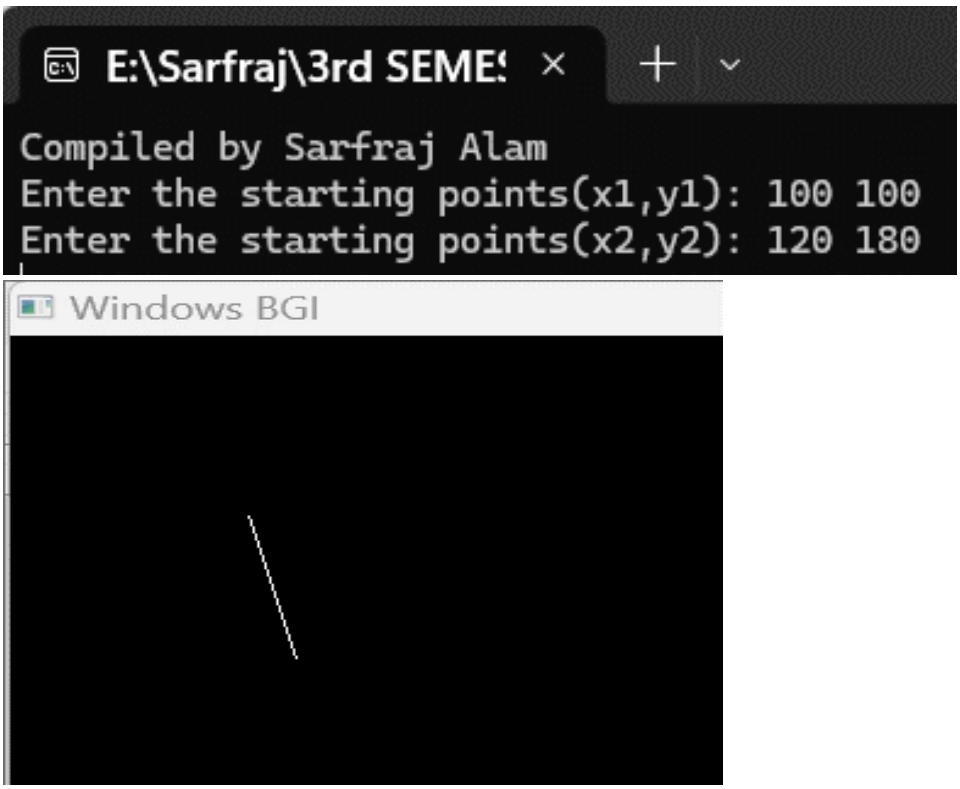
## Source code:

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

int i,dinc;

int main()

{

        printf("Compiled by Sarfraj Alam\n");

        int x1,x2,y1,y2,dx,dy;

        printf("Enter the starting points(x1,y1): ");

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

        printf("Enter the starting points(x2,y2): ");

        scanf("%d%d",&x2,&y2);

        dx=x2-x1;

        dy=y2-y1;

        float m=float(dy)/dx;

      dx=fabs(dx);

        dy=fabs(dy);

        float x=x1; float y=y1;

        int gm,gd=DETECT;

        initgraph(&gd,&gm,NULL);

        if(fabs(m)<1&&m>=0) {//for positive slope |m|<1

                for(i=1;i<=dx;i++)

                {

                        int x_inc=1;

                        putpixel(round(x),round(y),WHITE);

                        delay(10);

                        x=x+x_inc;

                        y=y+m*x_inc;

                }

        }
```

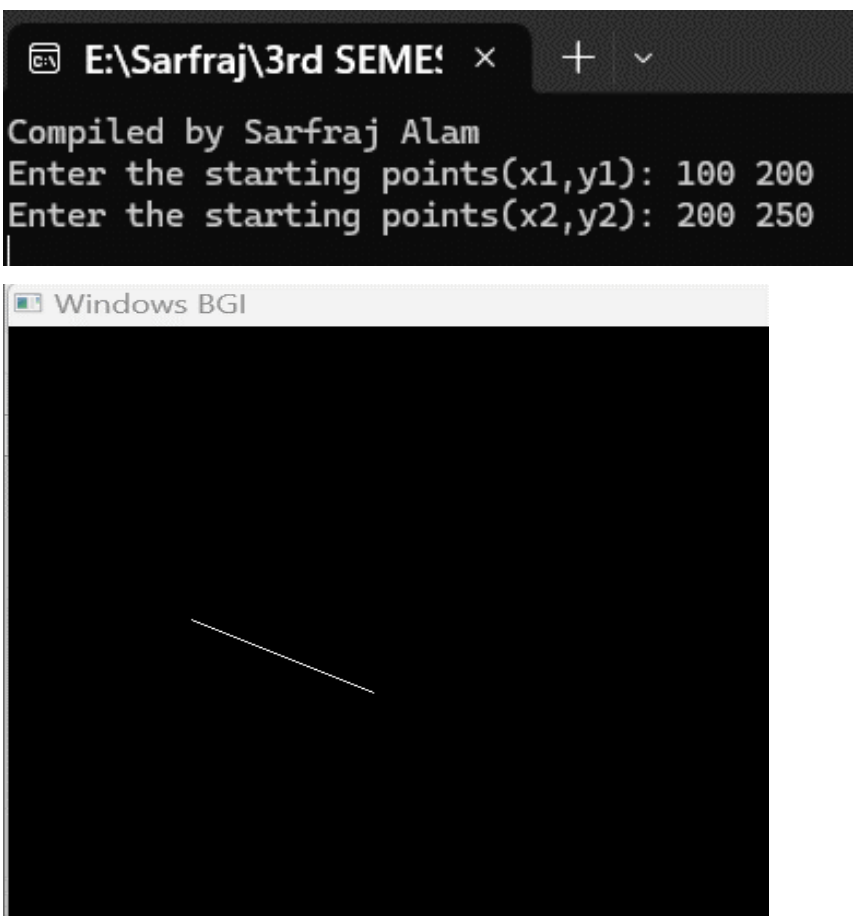
```
else if(fabs(m)>1&&m>=0) //for positive slope |m|>1
        {
                for(i=1;i<=dy;i++)
                {
                        int y_inc=1;
                        putpixel(round(x),round(y),WHITE);
                        delay(10);
                        x=x+(1/m);
                        y=y+y_inc;
                }
        }
        else if(fabs(m)>1&&m<0) //for negative slope |m|>1
        {
                for(i=1;i<=dy;i++)
                {
                        int y_inc=-1;
                        putpixel(round(x),round(y),WHITE);
                        delay(10);
                        x=x+(1/m);
                        y=y+y_inc;
                }
        }
        else
        {
        for(i=1;i<=dy;i++) //for negative slope |m|<1
                {
                        int x_inc=-1;
                        putpixel(round(x),round(y),WHITE);
                        delay(10);
                        x=x+x_inc;
                        y=y+m*x_inc;
                }
        }
        getch();
        closegraph();
}
```

**Output: DDA**

**For positive slope & |m|>1**

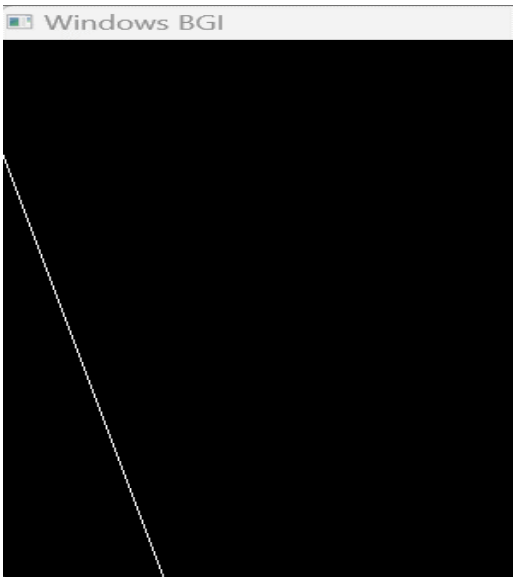



**For positive slope & |m|<1**



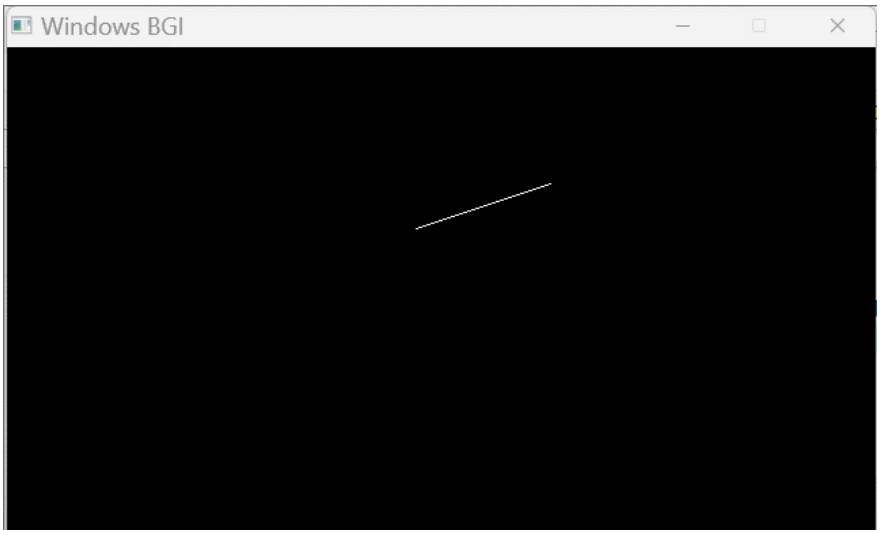
**For negative slope & |m|>1**

```
E:\Sarfraj\3rd SEME
Compiled by Sarfraj Alam
Enter the starting points(x1,y1): 100 500
Enter the starting points(x2,y2): 200 100
```




**For negative slope & |m|<1**

```
E:\Sarfraj\3rd SEME
Compiled by Sarfraj Alam
Enter the starting points(x1,y1): 400 100
Enter the starting points(x2,y2): 100 200
```



**2.WAP to implement BLA algorithm in C for:**

**1) |m|>1 positive slope**

**2) |m|<1 negative slope**

---

## Source code:

```
#include<stdio.h>

#include<graphics.h>

#include<math.h>

#include<conio.h>

int main()

{

        printf("Compiled by Sarfraj Alam\n");

        int gm,gd=DETECT;

        int x1,y1,x2,y2,i,j,Pk;

        float m,x,y;

        printf("Enter the initial coordinate(x1,y1): ");

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

        printf("Enter the final coordinate(x2,y2): ");

        scanf("%d%d",&x2,&y2);

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

        int dx=x2-x1;

        int dy=y2-y1;

        m=float(dy)/dx;

        dx=fabs(dx);

        dy=fabs(dy);

        if(fabs(m)>1) //for slope:|m|>1;

        {

        float P0=2*dx-dy;

                x=x1;

                y=y1;

                if(m>=0)// for positive slope & |m|>1

                {

                for(i=0;i<=dy;i++)

                {



        if(P0<0)
```

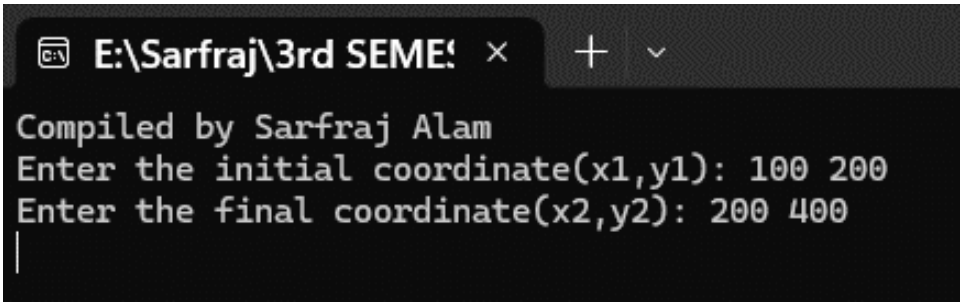
```
{
                x=x;
                y=y+1;
                putpixel(x,y,WHITE);
                delay(10);
                P0=P0+2*dx;
        }
else
{
                x=x+1;
                y=y+1;
                putpixel(x,y,WHITE);
                delay(10);
                P0=P0+2*dx-2*dy;
        }
}}
else
{
        for(i=0;i<=dy;i++)// for negative slope & |m|>1
        {
if(P0<0)
{
                x=x;
                y=y-1;
                putpixel(x,y,WHITE);
                delay(10);
                P0=P0+2*dx;
        }
else
{
                x=x+1;
                y=y-1;
                putpixel(x,y,WHITE);
                delay(10);
                P0=P0+2*dx-2*dy;
```

```
        }
        }
        }
}
else //for slope:|m|<1
        {
        float P0=2*dy-dx;
                x=x1;
                y=y1;
                if(m>=0)// for positive slope & |m|<1
                {
                for(i=0;i<=dx;i++)
                {

        if(P0<0)
        {
                        x=x+1;
                        y=y;
                        putpixel(x,y,WHITE);
                        delay(10);
                        P0=P0+2*dy;
                }
        else
        {
                        x=x+1;
                        y=y+1;
                        putpixel(x,y,WHITE);
                        delay(10);
                        P0=P0+2*dy-2*dx;
                }
        }}
        else
        {
                for(i=0;i<=dx;i++)// for negative slope & |m|<1
                {
        if(P0<0)
```

```
        {
                x=x+1;
                y=y-1;
                putpixel(x,y,WHITE);
                delay(10);
                P0=P0+2*dx;
            }
        else
        {
                x=x+1;
                y=y;
                putpixel(x,y,WHITE);
                delay(10);
                P0=P0+2*dx-2*dy;
            }
        }
        }
}
getch();
closegraph();
return 0;
}
```

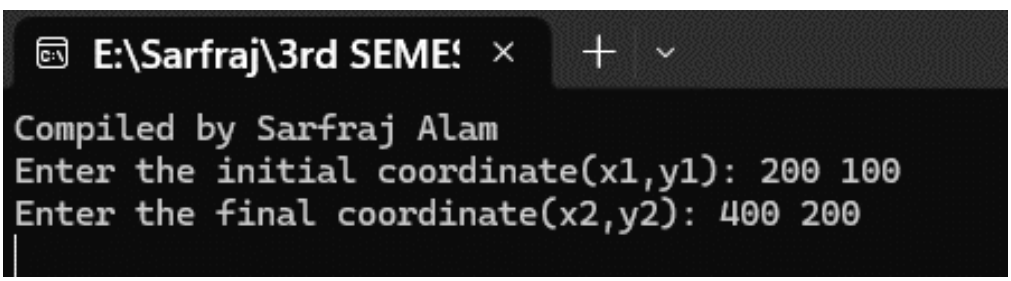
**Output: BLA**

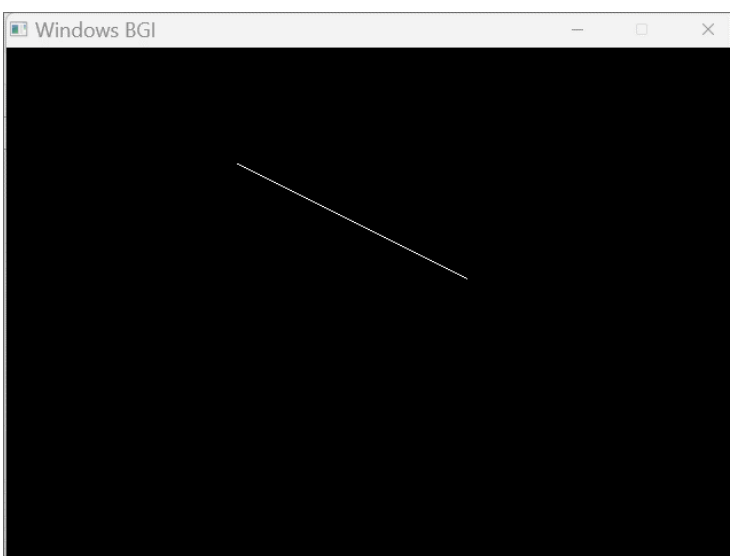
**For positive slope & |m|>1**



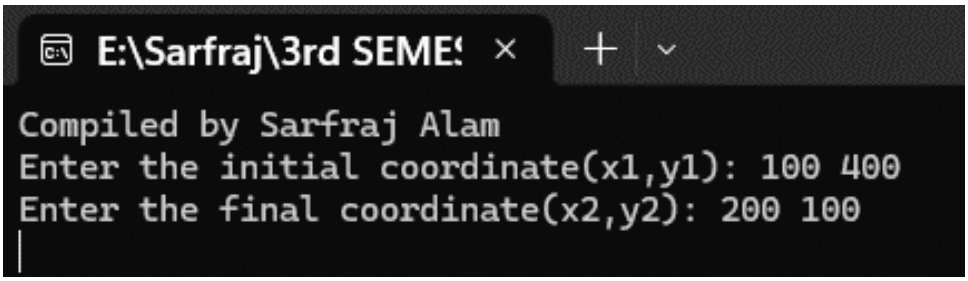


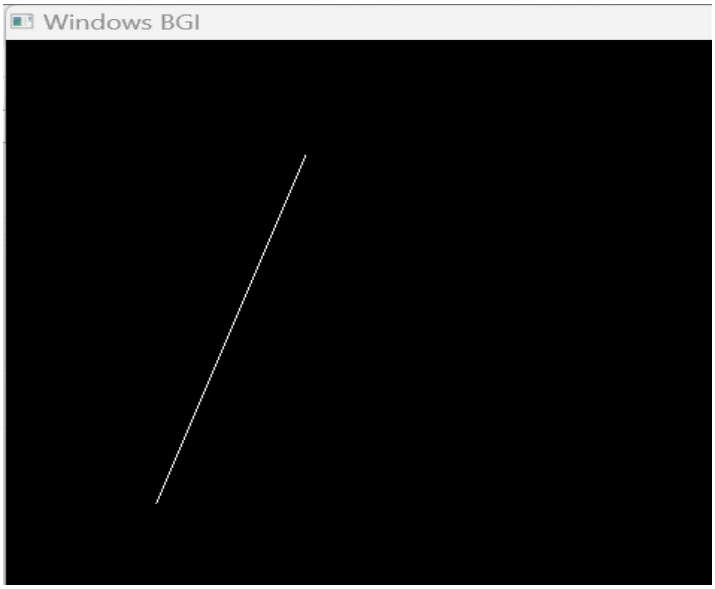
**For positive slope & |m|<1**



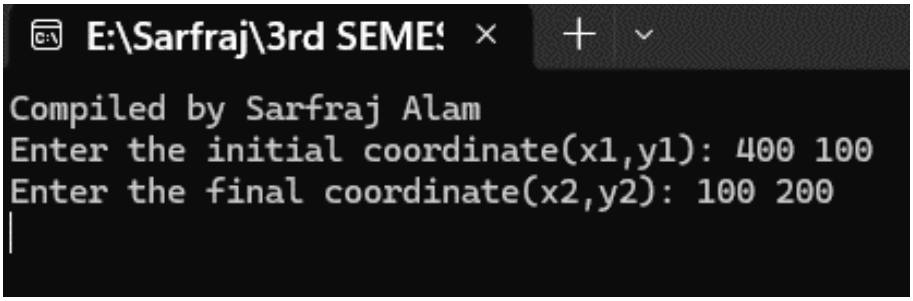



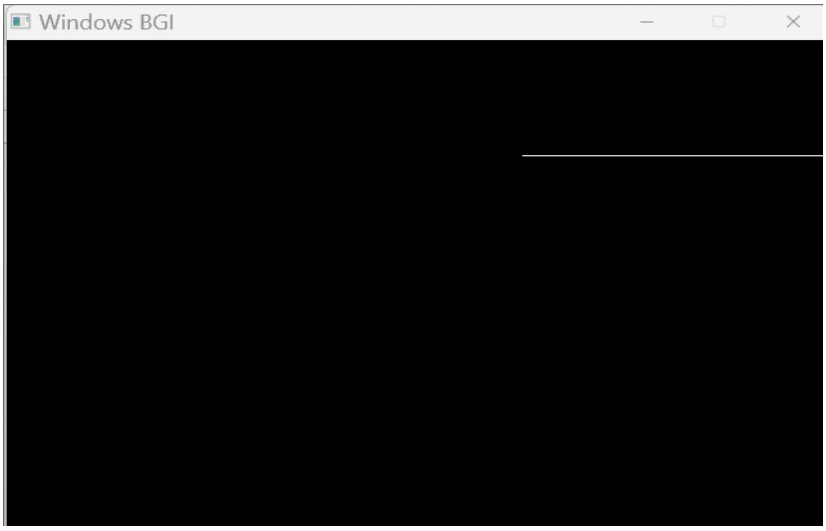
**For negative slope & |m|>1**




**For negative slope & |m|<1**



## 3.WAP to implement mid- point circle algorithm in C++.

**Source code:**

```
#include <iostream>

#include <graphics.h>

#include <conio.h> // For getch()


void drawCircle(int x0, int y0, int radius)

{

   int x = radius;

   int y = 0;

   int err = 0;


   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 (err <= 0)

      {

         y += 1;

         err += 2 * y + 1;

      }


      if (err > 0)

      {

         x -= 1;
```

```
    err -= 2 * x + 1;
      }
   }
}

int main()
{
   std::cout << "Compiled by Sarfraj Alam\n";
   int gdriver = DETECT, gmode, error, x, y, r;
   initgraph(&gdriver, &gmode," ");

   std::cout << "Enter radius of circle: ";
   std::cin >> r;

   std::cout << "Enter coordinates of center (x and y): ";
   std::cin >> x >> y;

   drawCircle(x, y, r);

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

**Output**

```
E:\Sarfraj\3rd SEME:

Compiled by Sarfraj Alam
Enter radius of circle: 100
Enter coordinates of center (x and y): 200 300
```

## 4.WAP to implement mid point ellipse algorithm in C++

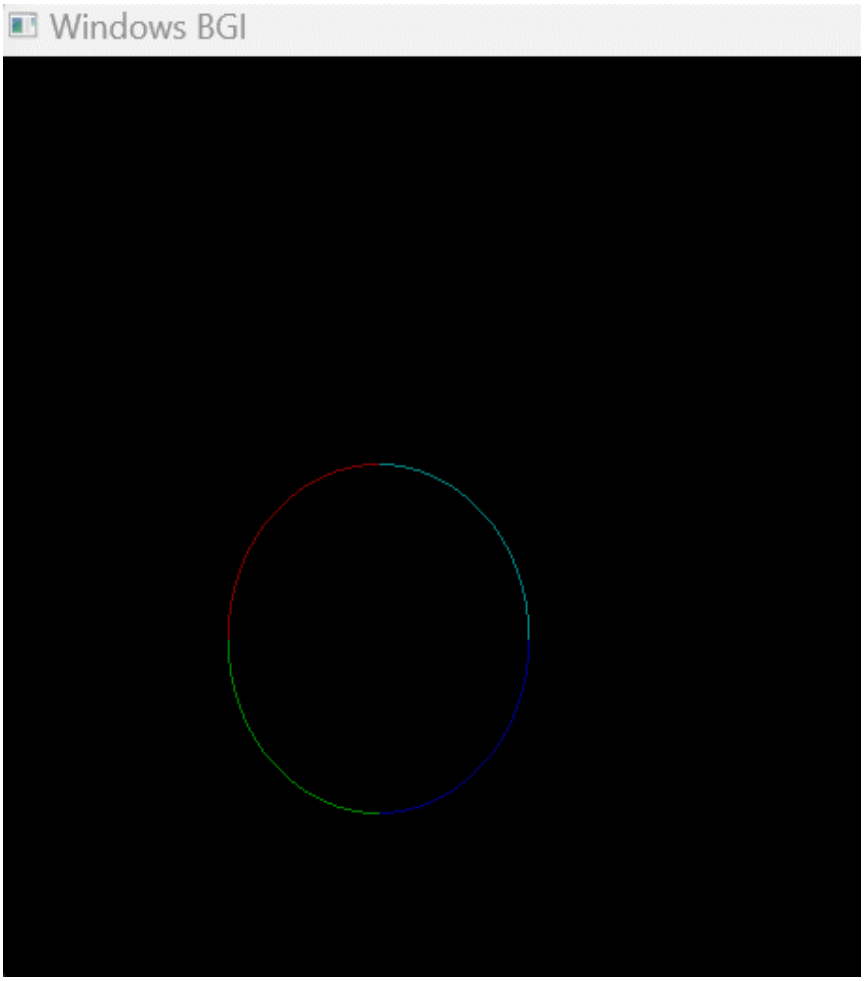
**Source code:**

```
#include <graphics.h>
#include <math.h>
#include <iostream>
using namespace std;
int main()
{
        cout << "Compiled by Sarfraj Alam\n";
  int gd = DETECT, gm;
  initgraph(&gd, &gm, (char *)"");
  int Xr, Yr, x1, y1, p, k = 0;
  cout << "Enter the x-radius of ellipse: ";
  cin >> Xr;
  cout << "Enter the y-radius of ellipse: ";
  cin >> Yr;
  cout << "Enter the centre coordinates of ellipse: ";
  cin >> x1 >> y1;
  p = pow(Yr, 2) - pow(Xr, 2) * Yr + 1 / 4 * pow(Xr, 2);
  int x = 0, y = Yr;
  while (2 * Yr * Yr * x < 2 * Xr * Xr * y)
  {
    putpixel(x + x1, y + y1, 1);
    putpixel(-x + x1, y + y1, 2);
    putpixel(x + x1, -y + y1, 3);
    putpixel(-x + x1, -y + y1, 4);
    if (p < 0)
    {
      x = x + 1;
      p = p + 2 * pow(Yr, 2) * x + pow(Yr, 2);
    }
    else
    {
      x = x + 1;
      y = y - 1;
      p = p + 2 * pow(Yr, 2) * x - 2 * pow(Xr, 2) * y + pow(Yr, 2);
    }
    delay(50);
  }

  p = Yr * Yr * (x + 1 / 2) * (x + 1 / 2) + Xr * Xr * (y - 1) * (y - 1) - Xr * Xr * Yr * Yr;
  while (y >= 0)
  {
```

```
      putpixel(x + x1, y + y1, 1);
      putpixel(-x + x1, y + y1, 2);
      putpixel(x + x1, -y + y1, 3);
      putpixel(-x + x1, -y + y1, 4);
      if (p > 0)
      {




        y = y - 1;
         p = p - 2 * pow(Xr, 2) * y + pow(Xr, 2);
      }
      else
      {
         x = x + 1;
         y = y - 1;
         p = p + 2 * pow(Yr, 2) * x - 2 * pow(Xr, 2) * y + pow(Xr, 2);
      }
      delay(50);
   }
   getch();
   closegraph();
}
```

## Output

```
E:\Sarfraj\3rd SEME!

Compiled by Sarfraj Alam
Enter the x-radius of ellipse: 80
Enter the y-radius of ellipse: 90
Enter the centre coordinates of ellipse: 200 300
```



**5.WAP to implement 2D transformation in C++**

1. **Translation**
2. **Rotation**
3. **Scaling**
4. **Reflection**
5. **Shearing**

---

**Source code:**

```
#include <iostream>

#include <graphics.h>

#include <math.h>

using namespace std;


void translation(int x1, int y1, int x2, int y2, int tx, int ty) {

   setcolor(3);

   rectangle(x1, y1, x2, y2);

   setcolor(13);

   rectangle(x1 + tx, y1 + ty, x2 + tx, y2 + ty);

   getch();

}

void rotation(int x1, int y1, int x2, int y2, double angle) {

   setcolor(3);

   rectangle(x1, y1, x2, y2);

   angle = (angle * 3.14) / 180;

   long xr = x1 + ((x2 - x1) * cos(angle) - (y2 - y1) * sin(angle));

   long yr = y1 + ((x2 - x1) * sin(angle) + (y2 - y1) * cos(angle));

   setcolor(13);

   rectangle(x1, y1, xr, yr);

   getch();

}

void scaling(int x1, int y1, int x2, int y2, int x, int y) {

   setcolor(3);

   rectangle(x1, y1, x2, y2);

   setcolor(13);
```

```
rectangle(x1 * x, y1 * y, x2 * x, y2 * y);
   getch();
}
void reflection(int x1, int y1, int x2, int y2, int x3, int y3) {
   setcolor(3);
   line(x1, y1, x2, y2);
   line(x1, y1, x3, y3);
   line(x2, y2, x3, y3);
   setcolor(13);
   line(x1, -y1 + 500, x2, -y2 + 500);
   line(x1, -y1 + 500, x3, -y3 + 500);
   line(x2, -y2 + 500, x3, -y3 + 500);
   getch();
}
void shearing(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4, int shx) {
   setcolor(3);
   line(x1, y1, x2, y2);
   line(x1, y1, x3, y3);
   line(x3, y3, x4, y4);
   line(x2, y2, x4, y4);
   x1 = x1 + shx * y1;
   x2 = x2 + shx * y2;
   x3 = x3 + shx * y3;
   x4 = x4 + shx * y4;
   setcolor(13);
   line(x1, y1, x2, y2);
   line(x1, y1, x3, y3);
   line(x3, y3, x4, y4);
   line(x2, y2, x4, y4);
   getch();
}
int main() {
        cout << "Compiled by Sarfraj Alam\n";
```

```
  int gd = DETECT, gm;
int s, x, y, x1, y1, x2, y2, x3, y3, x4, y4, tx, ty, shx;
  double angle;
  initgraph(&gd, &gm, NULL);

  setcolor(WHITE);
  line(0, getmaxy() / 2, getmaxx(), getmaxy() / 2);
  line(getmaxx() / 2, 0, getmaxx() / 2, getmaxy());

  cout << "Enter the choice of transformation:\n";
  cout << "1.Translation\n2.Rotation\n3.Scaling\n4.Reflection\n5.Shearing\n";
  cout << "\nSelection: ";
  cin >> s;

  switch (s) {
    case 1:
      cout << "Enter coordinates: ";
      cin >> x1 >> y1 >> x2 >> y2;
      cout << "Enter translation point: ";
      cin >> tx >> ty;
      translation(x1, y1, x2, y2, tx, ty);
      break;
    case 2:
      cout << "Enter coordinates: ";
      cin >> x1 >> y1 >> x2 >> y2;
      cout << "Enter Angle of rotation: ";
      cin >> angle;
      rotation(x1, y1, x2, y2, angle);
      break;
    case 3:
      cout << "Enter coordinates: ";
      cin >> x1 >> y1 >> x2 >> y2;
      cout << "Enter the scaling point: ";
```

```
        cin >> x >> y;
        scaling(x1, y1, x2, y2, x, y);
          break;
       case 4:
          cout << "Enter coordinates: ";
          cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
          reflection(x1, y1, x2, y2, x3, y3);
          break;
       case 5:
          cout << "Enter coordinates: ";
          cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3 >> x4 >> y4;
          cout << "Enter the shearing point: ";
          cin >> shx;
          shearing(x1, y1, x2, y2, x3, y3, x4, y4, shx);
          break;
       default:
          cout << "Invalid Selection\n";
          break;
    }
    closegraph();
}
```

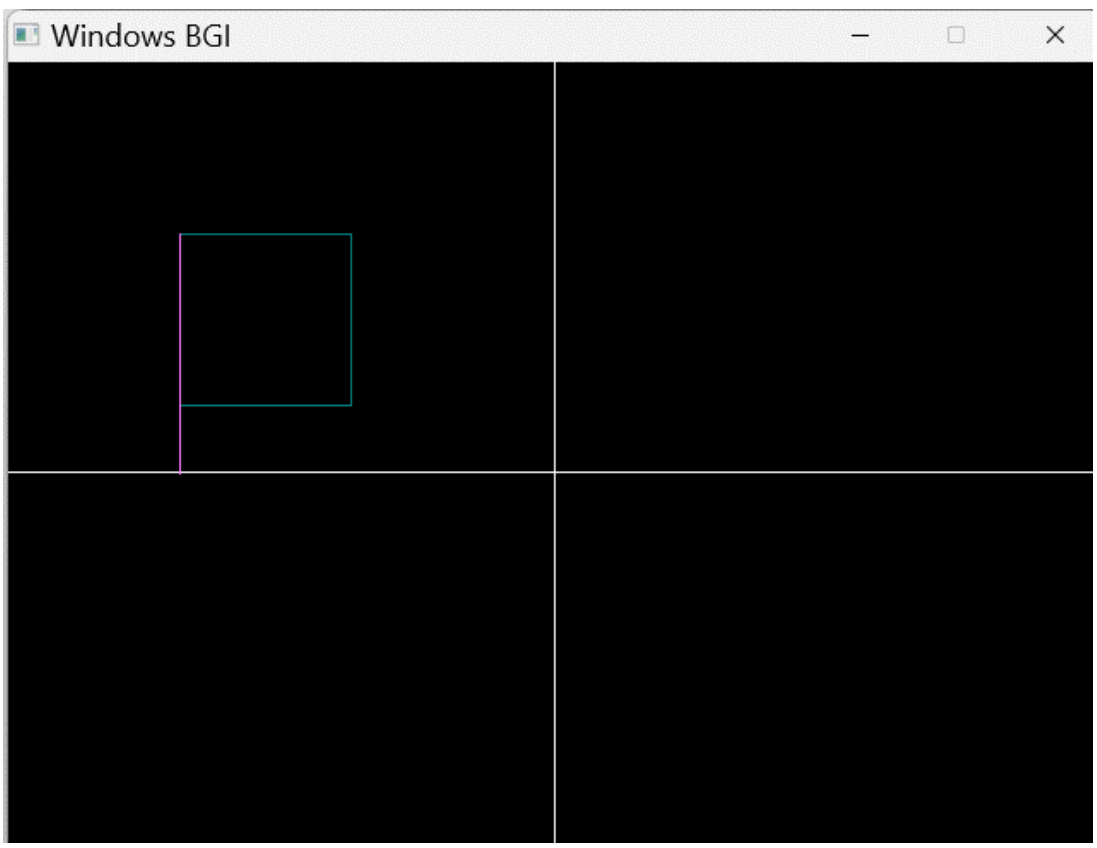
# Output

## For Translation

```
E:\Sarfraj\3rd SEME

Compiled by Sarfraj Alam
Enter the choice of transformation:
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing

Selection: 1
Enter coordinates: 100 100
140 140
Enter translation point: 80 80
```

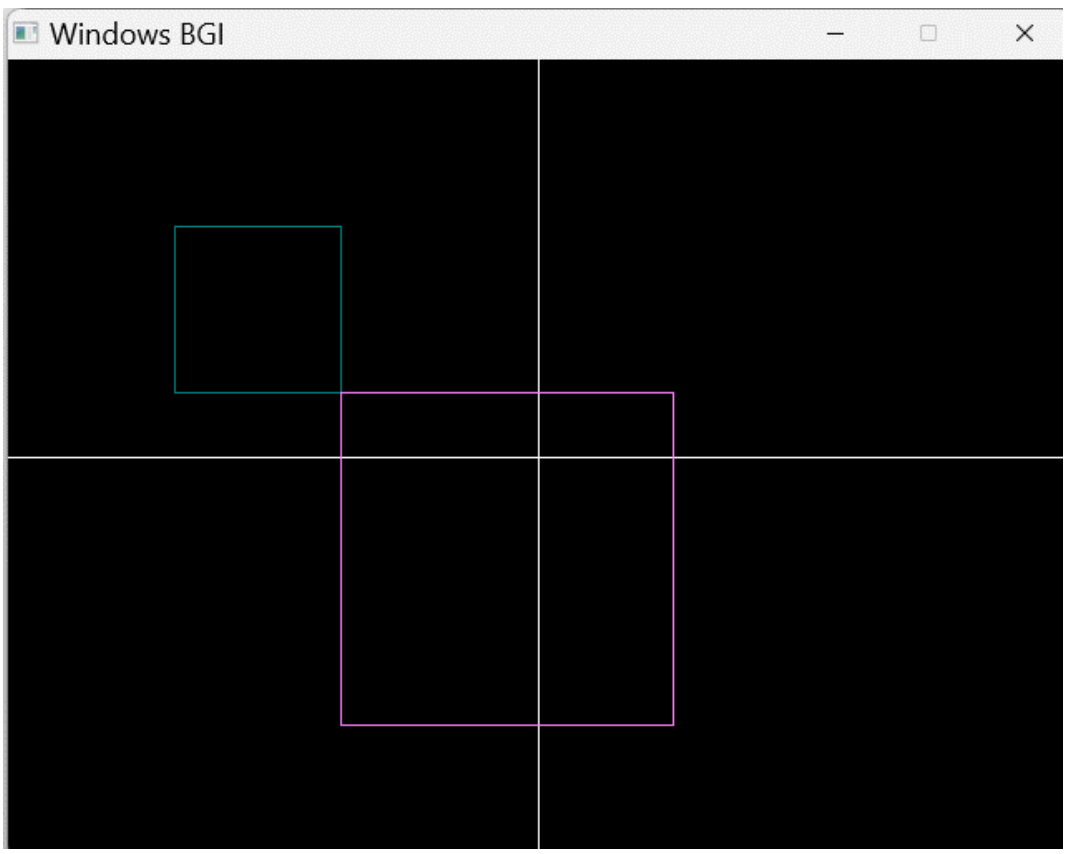
**For Rotation**

```
E:\Sarfraj\3rd SEME

Compiled by Sarfraj Alam
Enter the choice of transformation:
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing

Selection: 2
Enter coordinates: 100 100 200 200
Enter Angle of rotation: 45
```



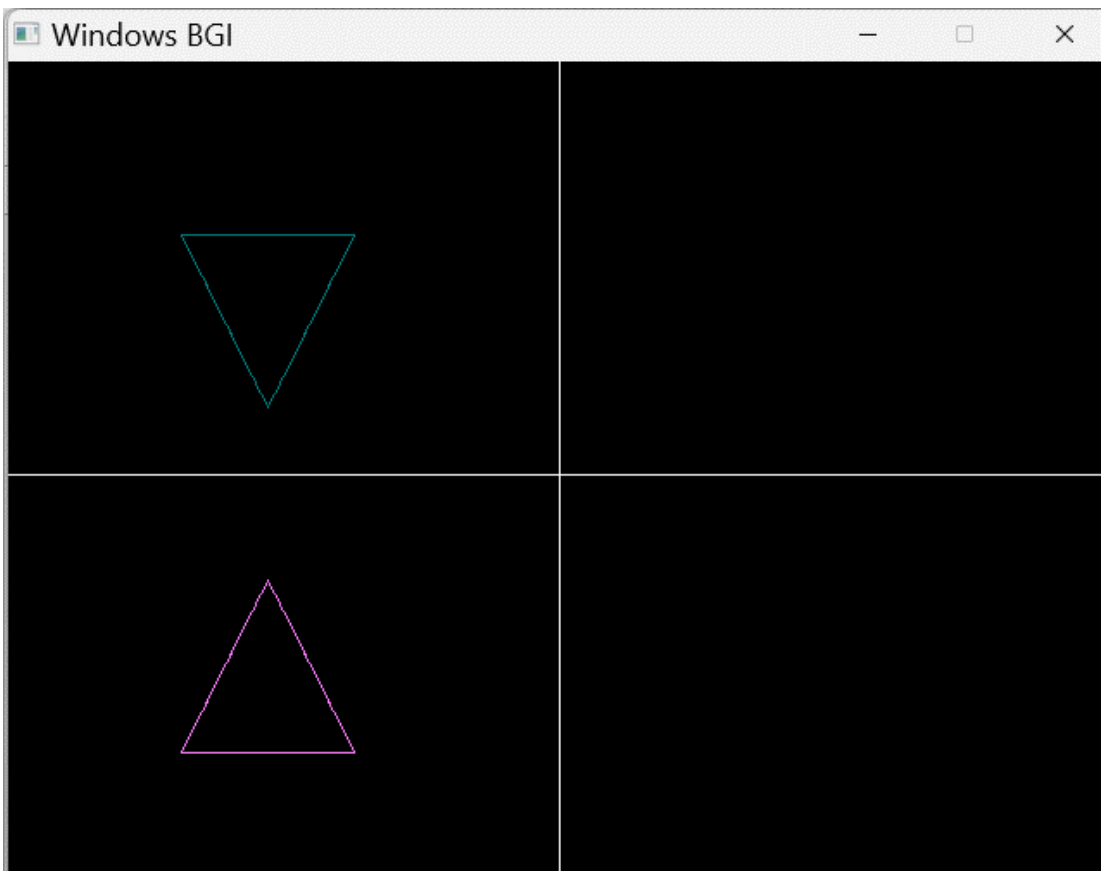
## For Scaling

```
E:\Sarfraj\3rd SEME

Compiled by Sarfraj Alam
Enter the choice of transformation:
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing

Selection: 3
Enter coordinates: 100 100
200 200
Enter the scaling point: 2 2
```



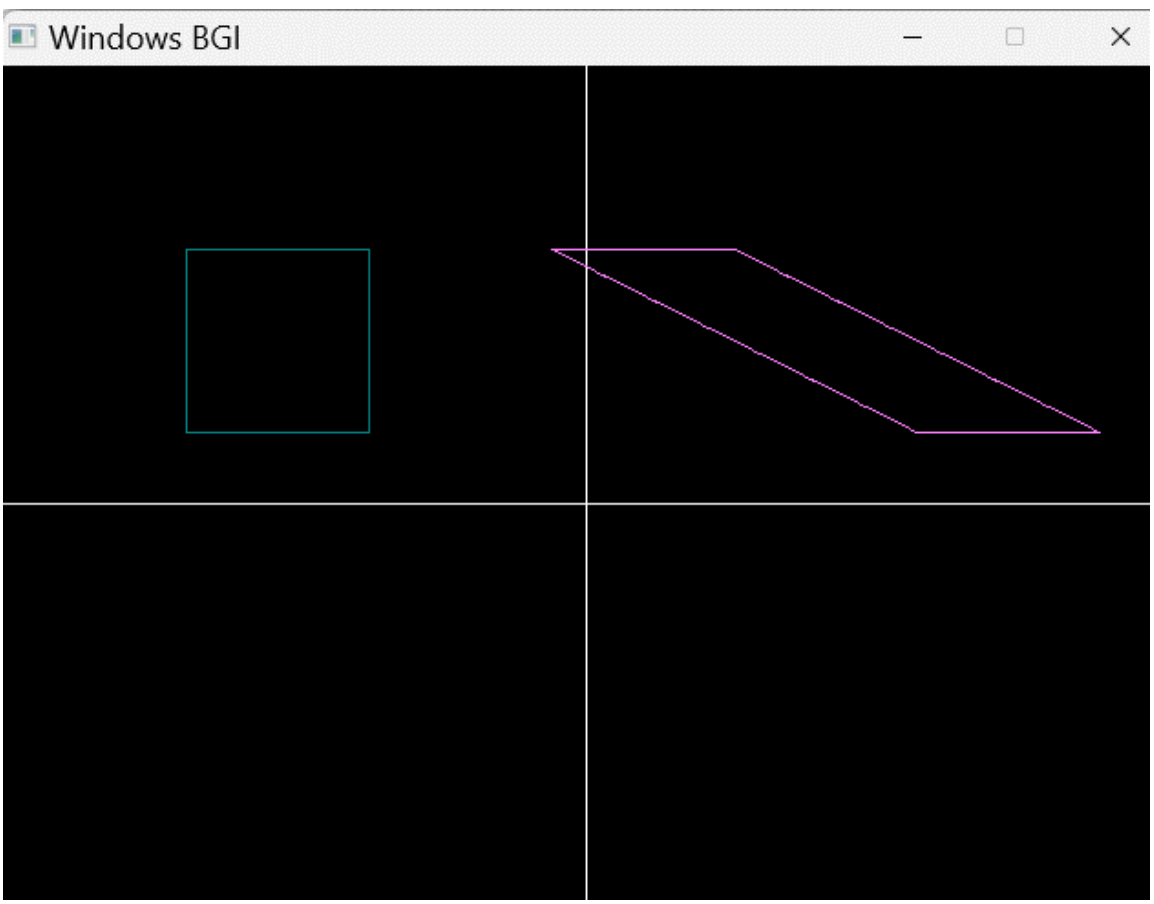
**For Reflection**

```
E:\Sarfraj\3rd SEME

Compiled by Sarfraj Alam
Enter the choice of transformation:
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing

Selection: 4
Enter coordinates: 100 100
200 100
150 200
```



**For Shearing**

```
E:\Sarfraj\3rd SEME

Compiled by Sarfraj Alam
Enter the choice of transformation:
1.Translation
2.Rotation
3.Scaling
4.Reflection
5.Shearing

Selection: 5
Enter coordinates: 100 100
200 100
100 200
200 200
Enter the shearing point: 2
```



## 6.WAP to implement 3D transformation in C++

### Source code:

```
#include <stdio.h>

#include <math.h>

// A simple 3D point with x, y, z coordinates

typedef struct {

   float x, y, z;

} Point3D;


// Display a 3D point

void showPoint(Point3D p) {

   printf("(%.1f, %.1f, %.1f)\n", p.x, p.y, p.z);

}

// Move the point by given amounts

Point3D movePoint(Point3D p, float move_x, float move_y, float move_z) {

   p.x += move_x;

   p.y += move_y;

   p.z += move_z;

   return p;

}

// Rotate point around X-axis by angle (in degrees)

Point3D turnX(Point3D p, float angle) {

   float rad = angle * (3.14159 / 180); // Convert to radians

   float new_y = p.y * cos(rad) - p.z * sin(rad);

   float new_z = p.y * sin(rad) + p.z * cos(rad);

   p.y = new_y;

   p.z = new_z;

   return p;

}

// Rotate point around Y-axis by angle (in degrees)

Point3D turnY(Point3D p, float angle) {

   float rad = angle * (3.14159 / 180);
```

```
  float new_x = p.x * cos(rad) + p.z * sin(rad);
  float new_z = -p.x * sin(rad) + p.z * cos(rad);
  p.x = new_x;
  p.z = new_z;
  return p;
}
// Rotate point around Z-axis by angle (in degrees)
Point3D turnZ(Point3D p, float angle) {
  float rad = angle * (3.14159 / 180);
  float new_x = p.x * cos(rad) - p.y * sin(rad);
  float new_y = p.x * sin(rad) + p.y * cos(rad);
  p.x = new_x;
  p.y = new_y;
  return p;
}

// Change size of point by given factors
Point3D resizePoint(Point3D p, float scale_x, float scale_y, float scale_z) {
  p.x *= scale_x;
  p.y *= scale_y;
  p.z *= scale_z;
  return p;
}

int main() {
  printf("Compiled by Sarfraj Alam\n");
  // Our starting point
  Point3D myPoint = {1.0, 1.0, 1.0};
  printf("Starting point: ");
  showPoint(myPoint);
  // Move the point
  myPoint = movePoint(myPoint, 2.0, 1.0, 0.5);
  printf("\nAfter moving by (2, 1, 0.5): ");
  showPoint(myPoint);
```

```
  // Rotate around X-axis
  myPoint = turnX(myPoint, 45);
  printf("\nAfter 45° X-rotation: ");
  showPoint(myPoint);
  // Rotate around Y-axis
  myPoint = turnY(myPoint, 30);
  printf("\nAfter 30° Y-rotation: ");
  showPoint(myPoint);
  // Change size
  myPoint = resizePoint(myPoint, 2.0, 0.5, 1.0);
  printf("\nAfter resizing by (2, 0.5, 1): ");
  showPoint(myPoint);
  return 0;
}
```

**Output**

```
E:\Sarfraj\3rd SEME

Compiled by Sarfraj Alam
Starting point: (1.0, 1.0, 1.0)

After moving by (2, 1, 0.5): (3.0, 2.0, 1.5)

After 45 X-rotation: (3.0, 0.4, 2.5)

After 30 Y-rotation: (3.8, 0.4, 0.6)

After resizing by (2, 0.5, 1): (7.7, 0.2, 0.6)

--------------------------------
Process exited after 0.09808 seconds with return value 0
Press any key to continue . . .
```