MADAN BHANDARI MEMORIAL COLLEGE

New Baneshor, Kathmandu

School of Science and Technology

Lab report of Computer Graphics

Submitted By:

Bibek Pathak
BSc.CSIT(3rd sem)
Section: A
Code: 2280601

Submitted To:

Rhishav Poudyal

Madan Bhandari Memorial College

S.N.	Title	Grade	Deadline	Sign
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				

Lab 1

Q. WAP in C to implement DDA algorithm for:

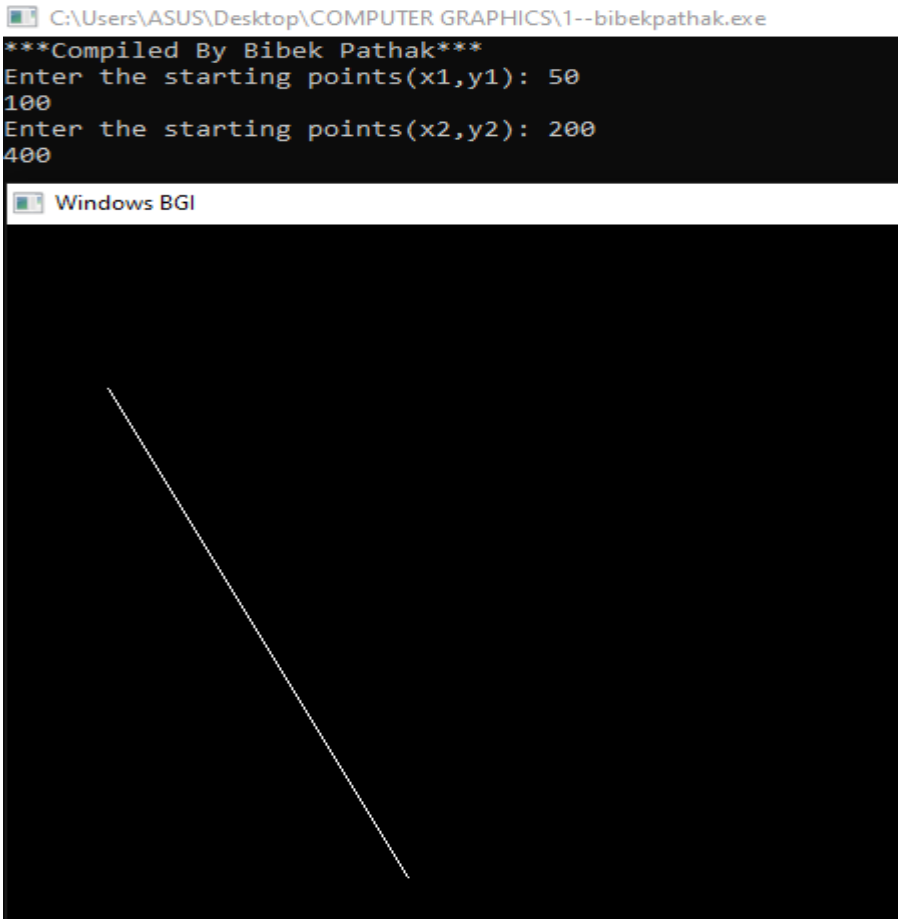
1. **|m|>1 positive slope**
2. **|m|>1 negative slope**

Source code:

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <math.h>
int i, dinc;
int main()
{
   printf("***Compiled By Bibek Pathak***\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
```

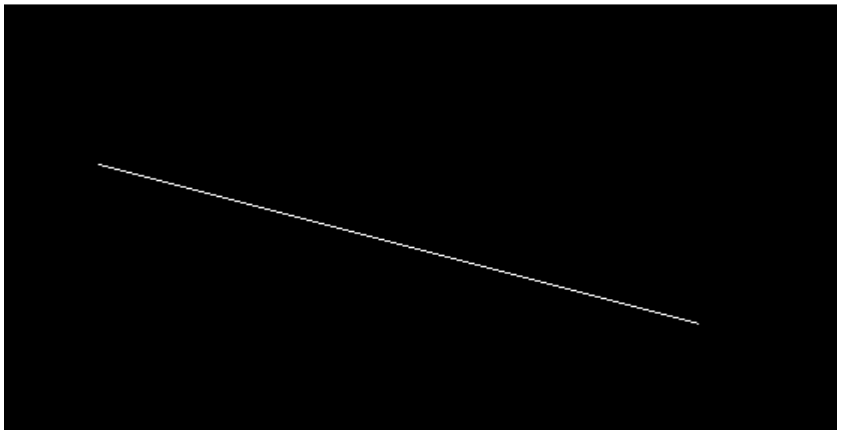
```
    {
        for (i = 1; i <= dy; i++)
        {
            int y_inc = -1;
            putpixel(round(x), round(y), WHITE);
            delay(10);
            x = x + (1 / m);
            y = y + y_inc;
        }
    }
    else
    {
        for (i = 1; i <= dy; i++) // for negative slope |m|<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:
For positive slope and |m|>1



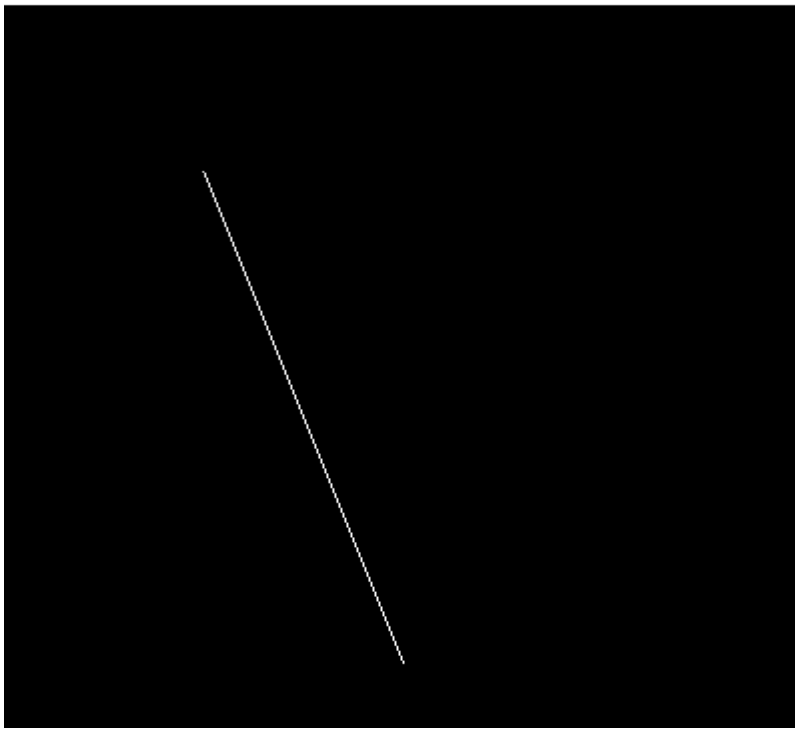
For positive slope and |m|<1

```
C:\Users\ASUS\Desktop\COMPUTER GRAPHICS\1--bibekpathak.exe
***Compiled By Bibek Pathak***
Enter the starting points(x1,y1): 50
100
Enter the starting points(x2,y2): 350
200
Windows BGI
```

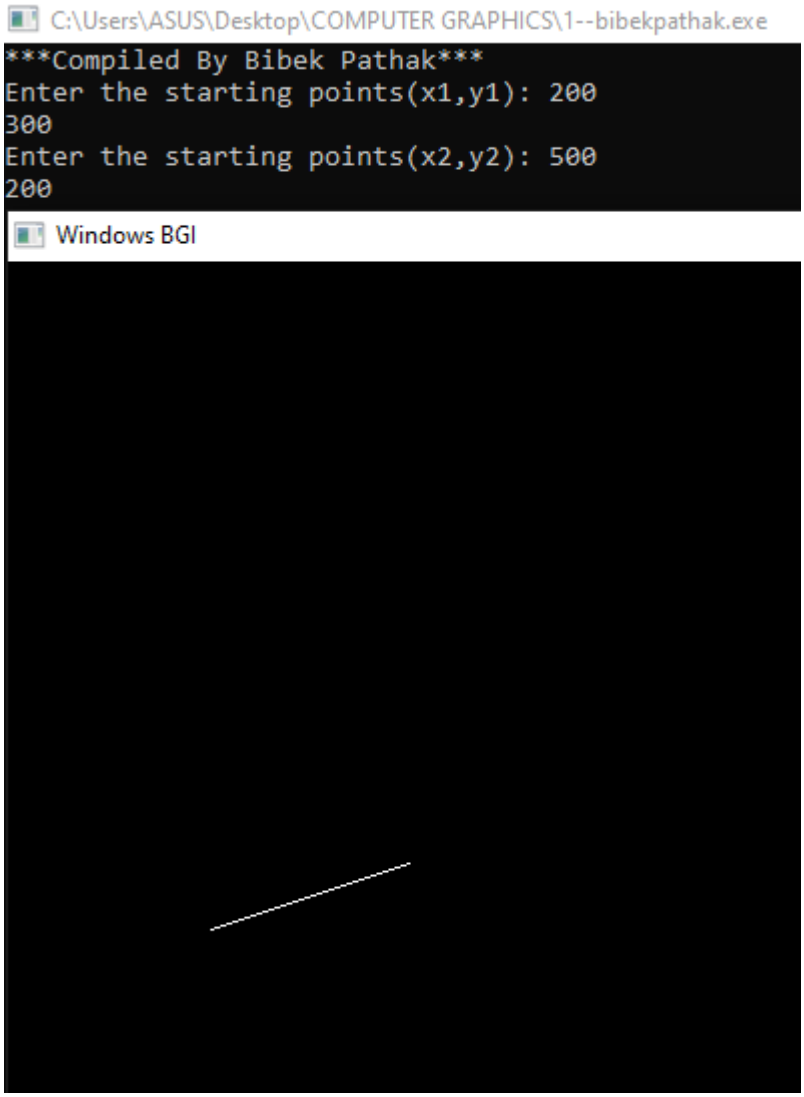


For negative slope and |m|>1

```
C:\Users\ASUS\Desktop\COMPUTER GRAPHICS\1--bibekpathak.exe
***Compiled By Bibek Pathak***
Enter the starting points(x1,y1): 200
400
Enter the starting points(x2,y2): 300
100
Windows BGI
```



For negative slope and |m|<1



Lab 2:

Q. WAP in C to implement BLA algorithm 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 Bibek Pathak***\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;
            }
```

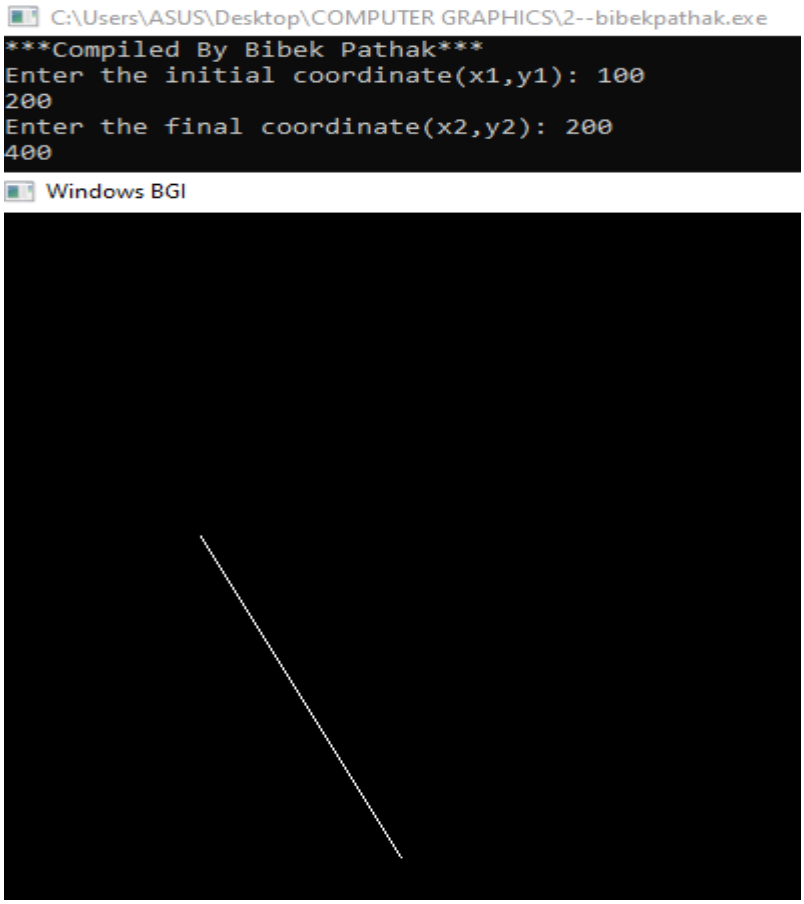
```
        }
    }
    else
    {
        for (i = 0; i <= dy; i++) // for negative slope & |m|>1
        {
            if (P0 < 0)
            {
                x = x;
                y = y - 1;
                putpixel(x, y, WHITE);
                delay(10);
                P0 = P0 + 2 * dx;
   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

For positive slope and |m|>1



For negative slope and |m|<1

C:\Users\ASUS\Desktop\COMPUTER GRAPHICS\2--bibekpathak.exe

```
***Compiled By Bibek Pathak***
Enter the initial coordinate(x1,y1): 200
300
Enter the final coordinate(x2,y2): 400
200
```

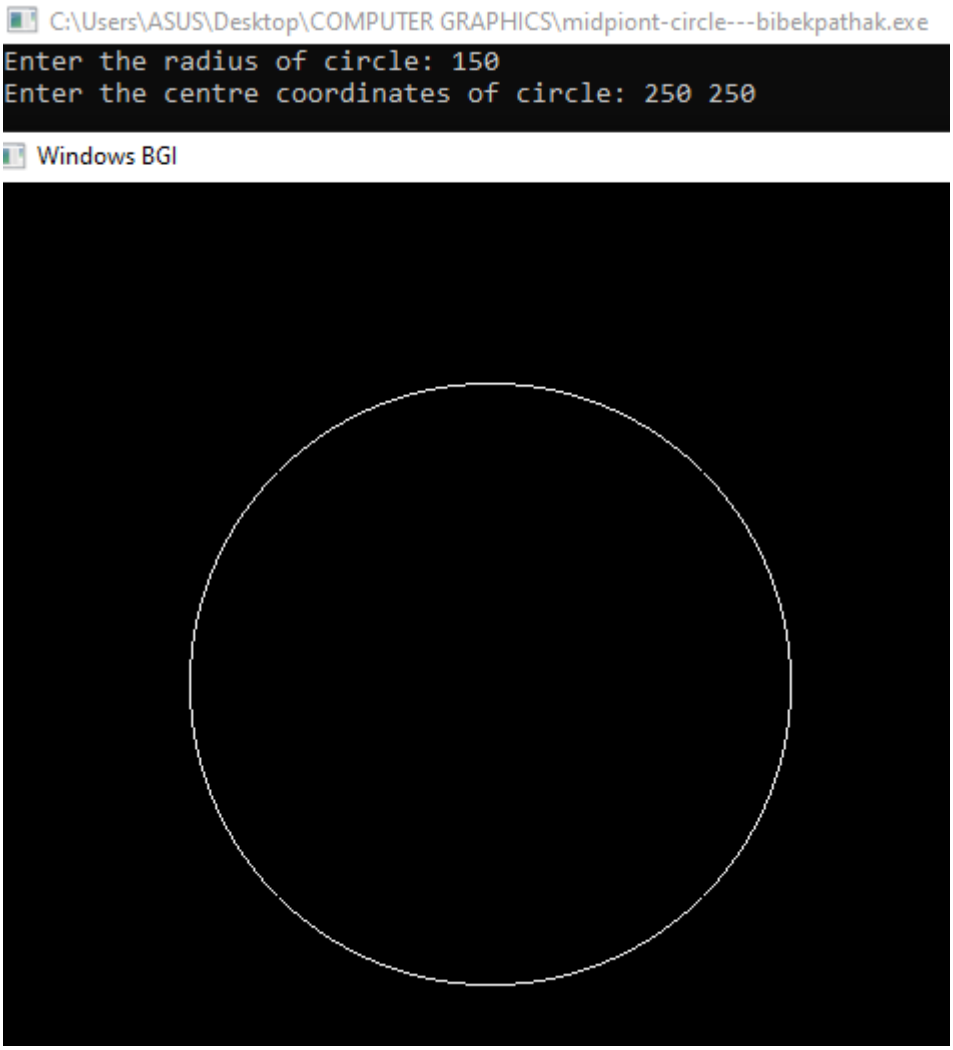
Windows BGI

Q. WAP in C to implement mid-point circle algorithm.

Source code:

```
#include <graphics.h>
#include <stdio.h>
int main()
{
   int gd = DETECT, gm;
   int radius, x1, y1, p, k = 0;
   printf("Enter the radius of circle: ");
   scanf("%d", &radius);
   printf("Enter the centre coordinates of circle: ");
   scanf("%d %d", &x1, &y1);
   initgraph(&gd, &gm, (char *)"");
   p = 5 / 4 - radius;
   int x = 0, y = radius;
   while (y > x)
   {
      putpixel(x + x1, y + y1, 15);
      putpixel(-x + x1, y + y1, 15);
      putpixel(x + x1, -y + y1, 15);
      putpixel(-x + x1, -y + y1, 15);
      putpixel(y + x1, x + y1, 15);
      putpixel(-y + x1, x + y1, 15);
      putpixel(y + x1, -x + y1, 15);
      putpixel(-y + x1, -x + y1, 15);
      if (p < 0)
      {
         x = x + 1;
         p = p + 2 * x + 1;
      }
      else
      {
         x = x + 1;
         y = y - 1;
         p = p + 2 * x + 1 - 2 * y;
      }
      delay(50);
   }
   getch();
   closegraph();
}
```

Output:


C:\Users\ASUS\Desktop\COMPUTER GRAPHICS\midpiont-circle---bibekpathak.exe
Enter the radius of circle: 150
Enter the centre coordinates of circle: 250 250
Windows BGI

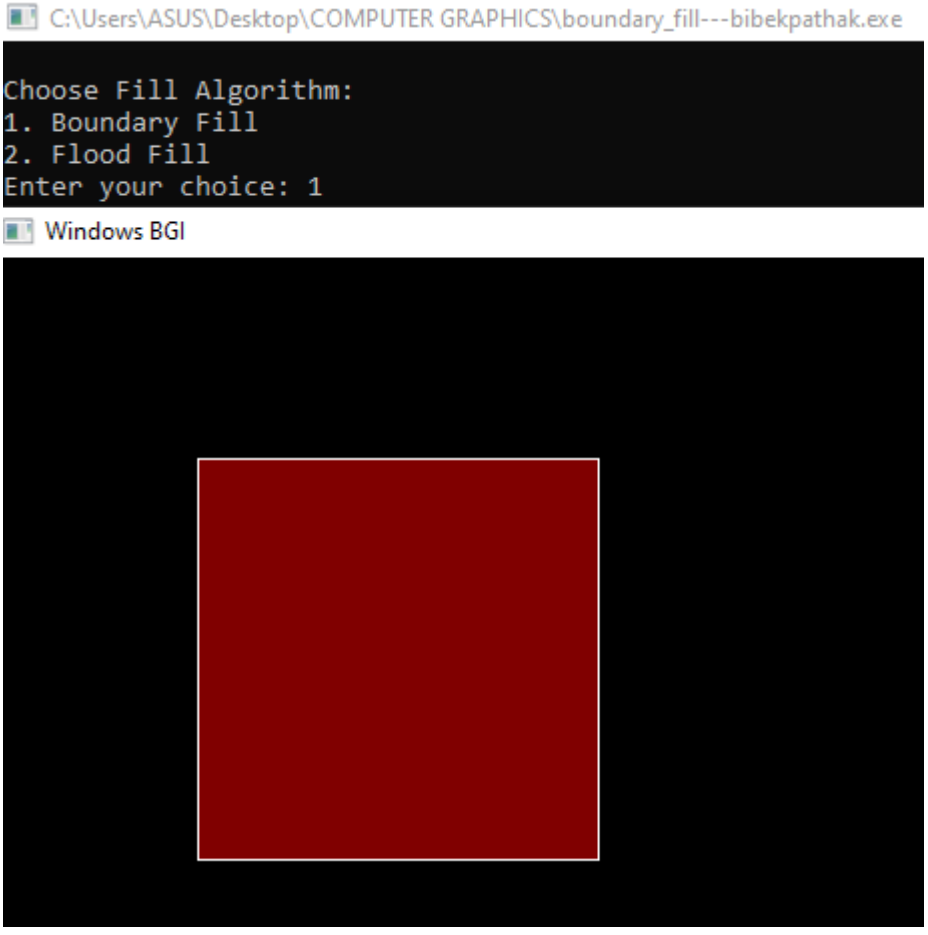
Q. WAP in C to implement boundary fill and flood fill algorithm.

Source code:

```
#include <graphics.h>
#include <dos.h>
#include <stdio.h>
#include <conio.h>
// Boundary Fill Algorithm
void boundaryFill(int x, int y, int fill_color, int boundary_color)
{
  int current = getpixel(x, y);
  if (current != boundary_color && current != fill_color)
  {
    putpixel(x, y, fill_color);
    delay(0.1);
    // 4-connected + diagonals (8-connected)
    boundaryFill(x + 1, y, fill_color, boundary_color);
    boundaryFill(x - 1, y, fill_color, boundary_color);
    boundaryFill(x, y + 1, fill_color, boundary_color);
    boundaryFill(x, y - 1, fill_color, boundary_color);
    boundaryFill(x + 1, y + 1, fill_color, boundary_color);
    boundaryFill(x - 1, y + 1, fill_color, boundary_color);
    boundaryFill(x + 1, y - 1, fill_color, boundary_color);
    boundaryFill(x - 1, y - 1, fill_color, boundary_color);
  }
}
// Flood Fill Algorithm
void floodFill(int x, int y, int new_color, int old_color)
{
  if (getpixel(x, y) == old_color)
  {
    putpixel(x, y, new_color);
    delay(0.1);
    // 4-connected + diagonals (8-connected)
    floodFill(x + 1, y, new_color, old_color);
    floodFill(x - 1, y, new_color, old_color);
    floodFill(x, y + 1, new_color, old_color);
    floodFill(x, y - 1, new_color, old_color);
    floodFill(x + 1, y + 1, new_color, old_color);
    floodFill(x - 1, y + 1, new_color, old_color);
    floodFill(x + 1, y - 1, new_color, old_color);
    floodFill(x - 1, y - 1, new_color, old_color);
  }
}
int main()
{
  int gd = DETECT, gm;
  initgraph(&gd, &gm, (char *)"");
  setcolor(WHITE);
  rectangle(100, 100, 300, 300); // drawing boundary
  int choice;
  printf("\nChoose Fill Algorithm:\n");
  printf("1. Boundary Fill\n");
  printf("2. Flood Fill\n");
  printf("Enter your choice: ");
```

```
    scanf("%d", &choice);
    switch (choice)
    {
    case 1:
        // fill with RED and boundary is WHITE
        boundaryFill(150, 150, RED, WHITE);
        break;
    case 2:
    {
        int x, y, old_color, new_color;
        printf("Enter seed point (x, y): ");
        scanf("%d %d", &x, &y);
        printf("Enter old color: ");
        scanf("%d", &old_color);
        printf("Enter new color: ");
        scanf("%d", &new_color);
        floodFill(x, y, new_color, old_color);
        break;
    }
    default:
        printf("Invalid choice!");
        break;
    }
    getch();
    closegraph();
    return 0;
}
```

Output:



Q. WAP in C to implement 2-D transformation.

Source code:

```
#include <graphics.h>
#include <stdio.h>
#include <math.h>
void display(int x1, int y1, int x2, int y2, int x3, int y3)
{
   int xmax = getmaxx();
   int ymax = getmaxy();
   int xmid = getmaxx() / 2;
   int ymid = getmaxy() / 2;
   // To draw vertical and horizontal line from mid of the screen
   line(xmid, 0, xmid, ymax);
   line(0, ymid, xmax, ymid);
   // To draw sides of the triangle
   line(x1 + xmid, y1 + ymid, x2 + xmid, y2 + ymid);
   line(x2 + xmid, y2 + ymid, x3 + xmid, y3 + ymid);
   line(x3 + xmid, y3 + ymid, x1 + xmid, y1 + ymid);
}
void translate(int x1, int y1, int x2, int y2, int x3, int y3, int tx, int ty)
{
   outtextxy(100, 100, "Before Translation:"); // display text at (x,y) coordinate
   display(x1, y1, x2, y2, x3, y3);
   delay(3000);
   cleardevice();
   outtextxy(100, 100, "After Translation:");
   display(x1 + tx, y1 + ty, x2 + tx, y2 + ty, x3 + tx, y3 + ty);
}
void scale(int x1, int y1, int x2, int y2, int x3, int y3, float sx, float sy)
{
   outtextxy(100, 100, "Before Scaling:");
   display(x1, y1, x2, y2, x3, y3);
   delay(3000);
   cleardevice();
   outtextxy(100, 100, "After Scaling:");
   display(x1 * sx, y1 * sy, x2 * sx, y2 * sy, x3 * sx, y3 * sy);
}
void arotate(int x1, int y1, int x2, int y2, int x3, int y3, int a) // anti-clock-wise rotation
{
   a = a * (3.1415 / 180);
   float c = cos(a);
   float s = sin(a);
   outtextxy(100, 100, "Before Rotation:");
   display(x1, y1, x2, y2, x3, y3);
   delay(3000);
   cleardevice();
   outtextxy(100, 100, "After Rotation:");
   display(x1 * c - y1 * s, x1 * s + y1 * c, x2 * c - y2 * s, x2 * s + y2 * c, x3 * c - y3 * s, x3 * s + y3 * c);
}
void crotate(int x1, int y1, int x2, int y2, int x3, int y3, int a) // clock-wise rotation
{
   a = a * (3.1415 / 180);
   float c = cos(a);
   float s = sin(a);
```

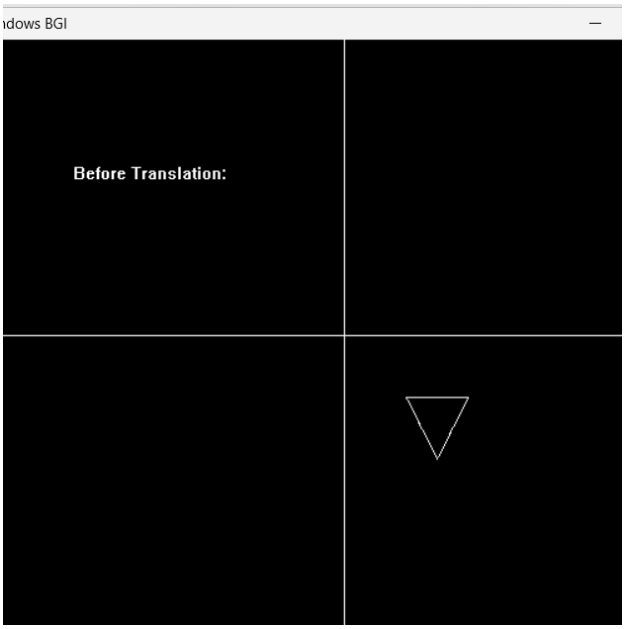
```
  outtextxy(100, 100, "Before Rotation:");
  display(x1, y1, x2, y2, x3, y3);
  delay(3000);
  cleardevice();
  outtextxy(100, 100, "After Rotation:");
  display(x1 * c + y1 * s, -x1 * s + y1 * c, x2 * c + y2 * s, -x2 * s + y2 * c, x3 * c + y3 * s, -x3 * s + y3 * c);
}
void xreflect(int x1, int y1, int x2, int y2, int x3, int y3)
{
  outtextxy(100, 100, "Before Reflection:");
  display(x1, y1, x2, y2, x3, y3);
  delay(3000);
  cleardevice();
  outtextxy(100, 100, "After Reflection about x-axis:");
  display(x1, -y1, x2, -y2, x3, -y3);
}
void yreflect(int x1, int y1, int x2, int y2, int x3, int y3)
{
  outtextxy(100, 100, "Before Reflection:");
  display(x1, y1, x2, y2, x3, y3);
  delay(3000);
  cleardevice();
  outtextxy(100, 100, "After Reflection about x-axis:");
  display(-x1, y1, -x2, y2, -x3, y3);
}
void xshear(int x1, int y1, int x2, int y2, int x3, int y3, float shx)
{
  outtextxy(100, 100, "Before Shearing:");
  display(x1, y1, x2, y2, x3, y3);
  delay(3000);
  cleardevice();
  outtextxy(100, 100, "After Shearing about x-axis:");
  display(x1 + shx * y1, y1, x2 + shx * y2, y2, x3 + shx * y3, y3);
}
void yshear(int x1, int y1, int x2, int y2, int x3, int y3, float shy)
{
  outtextxy(100, 100, "Before Shearing:");
  display(x1, y1, x2, y2, x3, y3);
  delay(3000);
  cleardevice();
  outtextxy(100, 100, "After Shearing about y-axis:");
  display(x1, y1 + shy * x1, x2, y2 + shy * x2, x3, y3 + shy * x3);
}
void xyshear(int x1, int y1, int x2, int y2, int x3, int y3, float shx, float shy)
{
  outtextxy(100, 100, "Before Shearing:");
  display(x1, y1, x2, y2, x3, y3);
  delay(3000);
  cleardevice();
  outtextxy(100, 100, "After Shearing about xy-axis:");
  display(x1 + shx * y1, y1 + shy * x1, x2 + shx * y2, y2 + shy * x2, x3 + shx * y3, y3 + shy * x3);
}
int main()
{
  int x1, y1, x2, y2, x3, y3;
```

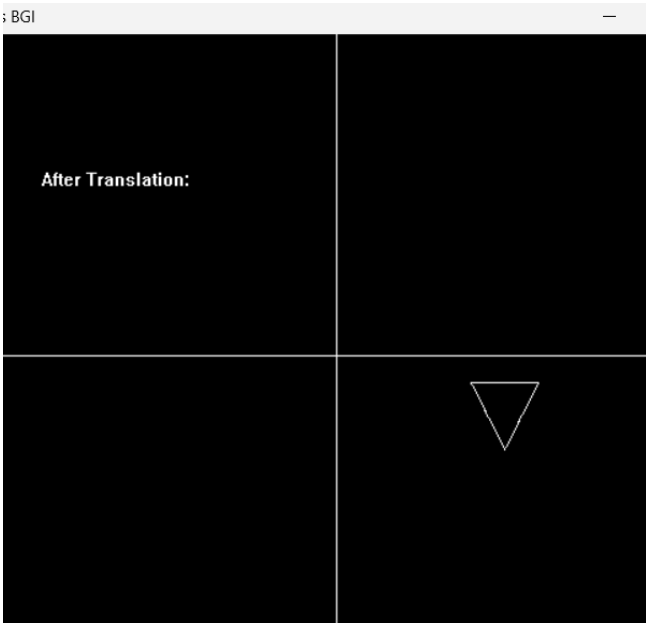
```
int gd = DETECT, gm;
printf("Enter the co-ordinates of the triangle: x1, y1, x2, y2, x3, y3:\n");
scanf("%d %d %d %d %d %d", &x1, &y1, &x2, &y2, &x3, &y3);
while (1)
{
  int ch;
  printf("Enter:\n"
      "1. For Translation\n"
      "2. For Scaling\n"
      "3. For Anticlockwise Rotation\n"
      "4. For Clockwise Rotation\n"
      "5. For Reflection about x-axis\n"
      "6. For Reflection about y-axis\n"
      "7. For Shearing about x-axis\n"
      "8. For Shearing about y-axis\n"
      "9. For Shearing about xy-axis\n"
      "10. For Exit\n");
  scanf("%d", &ch);
  if (ch == 1)
  {
    int tx, ty;
    printf("Enter Translation Factors tx and ty: \n");
    scanf("%d %d", &tx, &ty);
    initgraph(&gd, &gm, NULL);
    translate(x1, y1, x2, y2, x3, y3, tx, ty);
    getch();
    closegraph();
  }
  if (ch == 2)
  {
    float sx, sy;
    printf("Enter Scaling Factors sx and sy: \n");
    scanf("%f %f", &sx, &sy);
    initgraph(&gd, &gm, NULL);
    scale(x1, y1, x2, y2, x3, y3, sx, sy);
    getch();
    closegraph();
  }
  if (ch == 3)
  {
    float a;
    printf("Enter Rotation angle: \n");
    scanf("%f", &a);
    initgraph(&gd, &gm, NULL);
    arotate(x1, y1, x2, y2, x3, y3, a);
    getch();
    closegraph();
  }
  if (ch == 4)
  {
    float a;
    printf("Enter Rotation angle: \n");
    scanf("%f", &a);
    initgraph(&gd, &gm, NULL);
    crotate(x1, y1, x2, y2, x3, y3, a);
```

```
            getch();
            closegraph();
        }
        if (ch == 5)
        {
            initgraph(&gd, &gm, NULL);
            xreflect(x1, y1, x2, y2, x3, y3);
            getch();
            closegraph();
        }
        if (ch == 6)
        {
            initgraph(&gd, &gm, NULL);
            yreflect(x1, y1, x2, y2, x3, y3);
            getch();
            closegraph();
        }
        if (ch == 7)
        {
            float shx;
            printf("Enter Shearing factor shx: \n");
            scanf("%f", &shx);
            initgraph(&gd, &gm, NULL);
            xshear(x1, y1, x2, y2, x3, y3, shx);
            getch();
            closegraph();
        }
        if (ch == 8)
        {
            float shy;
            printf("Enter Shearing factor shy: \n");
            scanf("%f", &shy);
            initgraph(&gd, &gm, NULL);
            yshear(x1, y1, x2, y2, x3, y3, shy);
            getch();
            closegraph();
        }
        if (ch == 9)
        {
            float shx, shy;
            printf("Enter Shearing factors shx and shy: \n");
            scanf("%f %f", &shx, &shy);
            initgraph(&gd, &gm, NULL);
            xyshear(x1, y1, x2, y2, x3, y3, shx, shy);
            getch();
            closegraph();
        }
        if (ch == 0)
        {
            printf("EXITED\n");
            break;
        }
    }
    return 0;
}
```

Output: Translation:

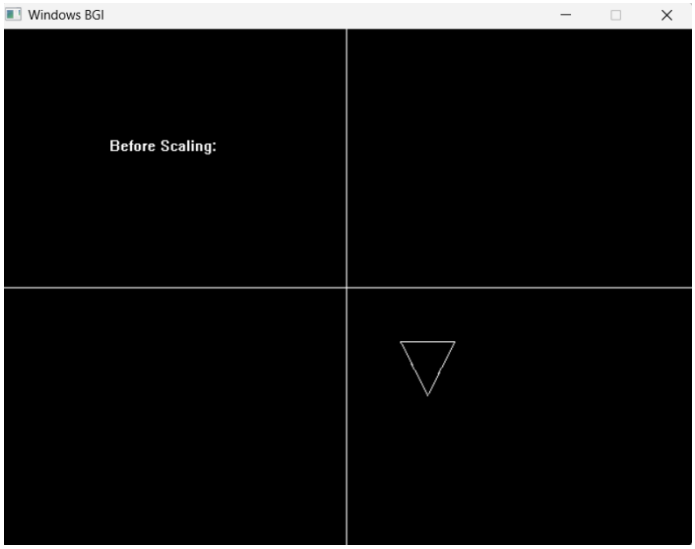
```
C:\Users\ASUS\Desktop\COMPUTER GRAPHICS\2-d--bibekpathak.exe
Enter the co-ordinates of the triangle: x1, y1, x2, y2, x3, y3:
50 50 100 50 75 100
Enter:
1. For Translation
2. For Scaling
3. For Anticlockwise Rotation
4. For Clockwise Rotation
5. For Reflection about x-axis
6. For Reflection about y-axis
7. For Shearing about x-axis
8. For Shearing about y-axis
9. For Shearing about xy-axis
10. For Exit
1
Enter Translation Factors tx and ty:
50 -30
```

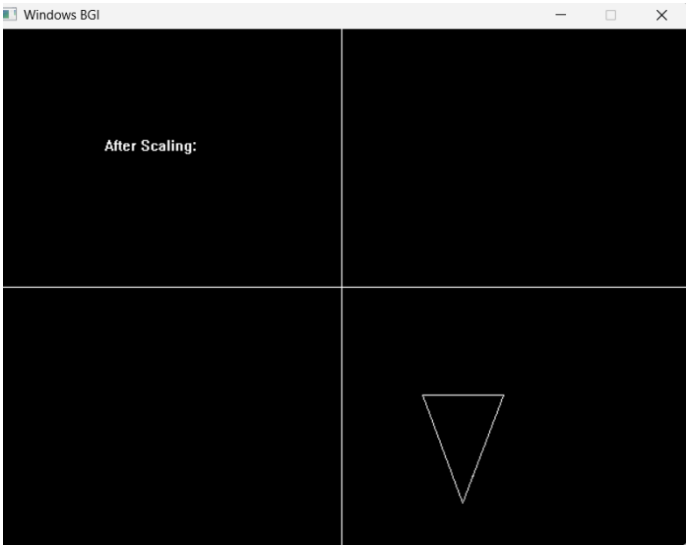




Scaling

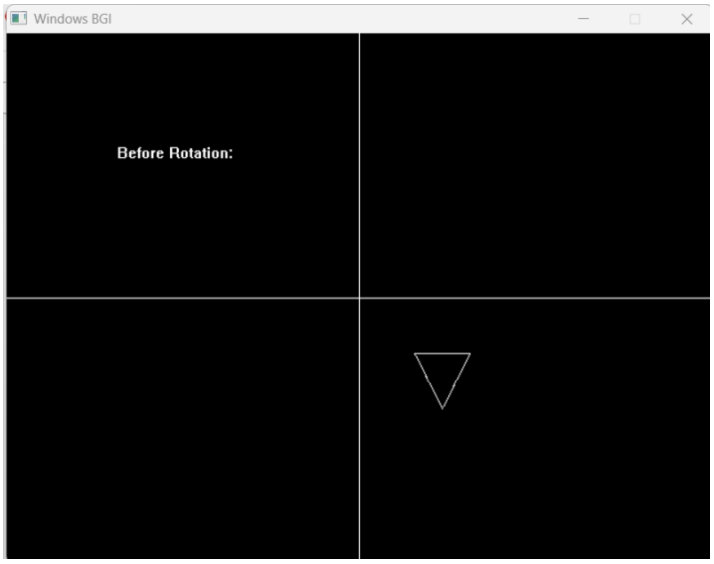
```
2
Enter Scaling Factors sx and sy:
1.5     2.0
```

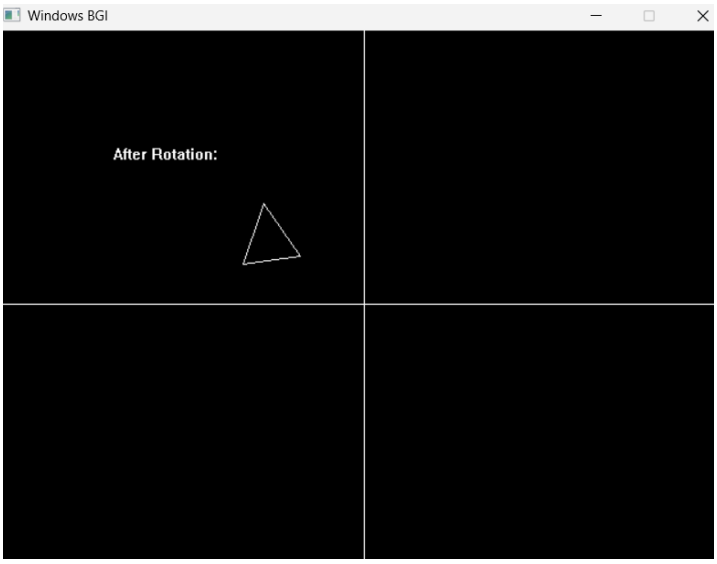




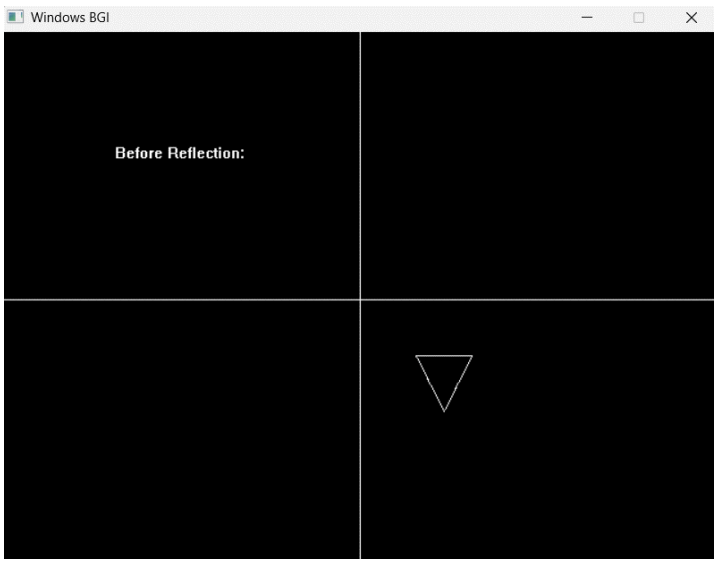
Rotation:

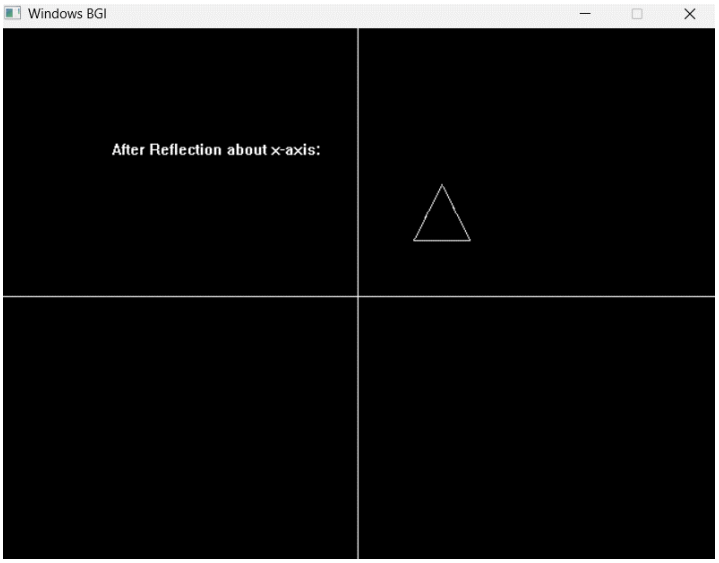
```
3
Enter Rotation angle:
180
```





Reflection:





Shearing:

```
Enter Shearing factors shx and shy:
1.0     0.5
```