CG Experiment 7.

Yash Sarang 47/D6AD

Aim: To implement 2D transformation using transition, rotation, scaling, reflection & shear.

Theory:

Translation: A translation process moves every point a constant distance in a specified direction which can be described as a rigid motion.

Suppose, if pt $f(x,y)$ is to be translated by amount D_x and D_y to pt $f(x',y')$ then.

$$x' = D_x + x$$
$$y' = D_y + y.$$

or $P' = P + T$

where,

$P = (x,y)$; $P'(x',y')$ and $T' = (D_x, D_y)$

and T is known as translation factor.

- Scaling: A scaling transformation alters the size of the object.
  In the scaling operation, we either compress or expand the dimension of the object.

  Scaling operation can be achieved by multiplying each vertex coordinate (x, y) of the polygon by scaling factor S_x and S_y to produce the transformation coordinates as (x', y')

$$\therefore x' = S_x \cdot x$$
$$\text{and } y' = S_y \cdot y.$$

Matrix form of Scaling:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

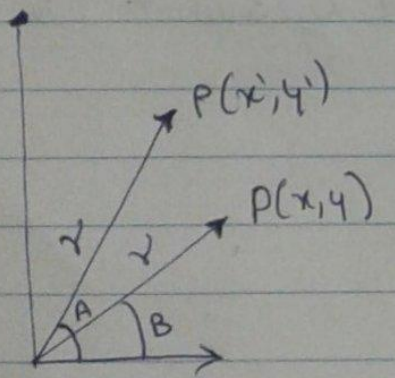
- Rotation:

A rotation is a process that rotates an object by a given angle about a given pivot point.

In order to rotate an object, we need to rotate each vertex of the figure individually.

On rotating a point $P(x,y)$ by an angle A about the origin we get a point $P(x',y')$.

x' and y' can be calculated as follows:



We know that,

$x = r\cos B$, $y = r\sin B$.

$r' = r\cos(A+B)$

$= r\cos A\cos B - r\sin A\sin B$.

$\therefore x' = x\cos A - y\sin B$

& $y' = x\sin A + y\cos A$

- Shear: A transformation that slants the slope of an object is called the shear transformation.

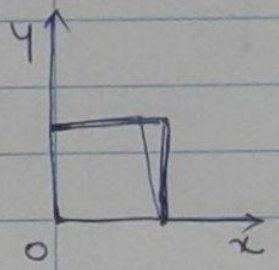
Shearing is also known as Skewy.

There are two types of shear transformation:

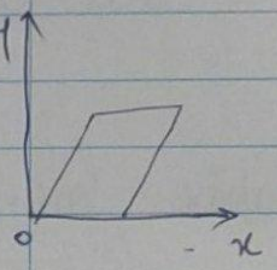
(i) x- shear
(ii) y- shear.

(i) x-shear : The x-shear preserves the y-coordinates and changes are made to the x-coordinates causing vertical lines to tilt in right or left direction

Example



(a) Original object



(b) object after x shear

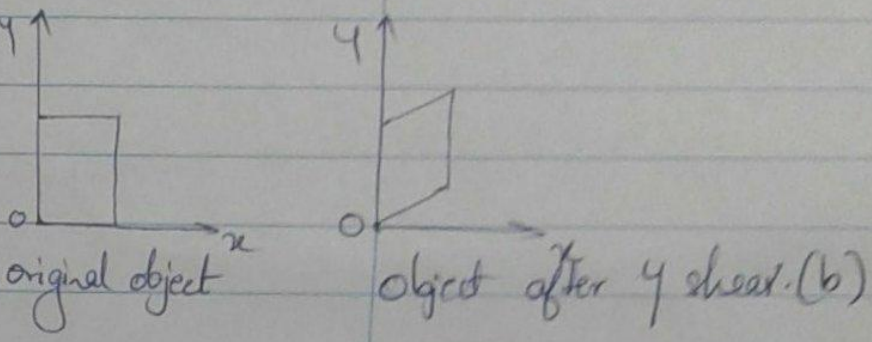
x shear transformation matrix:

$$X_{shear} = \begin{bmatrix} 1 & shx & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

$$y' = y + shx \cdot x$$
$$x' = x.$$

(ii) y-shear : The y shear preserves the x coordinates and changes are made to the y coordinates causing horizontal lines to tilt upwards or downwards.

y shear example:

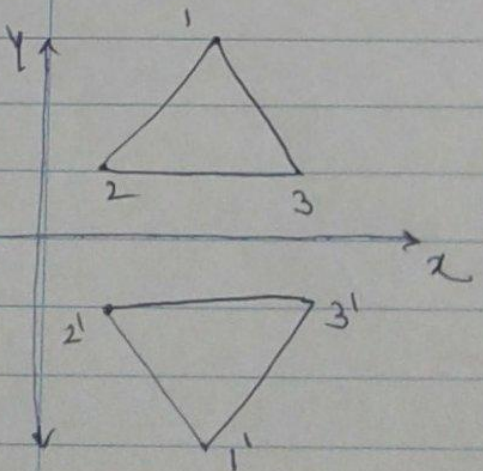


(a) original object object after y shear. (b)

- Reflection : Reflection is the mirror image of the object. In other words, the reflection process is a rotation operation with 180°.

In reflection, the size of object does not change.

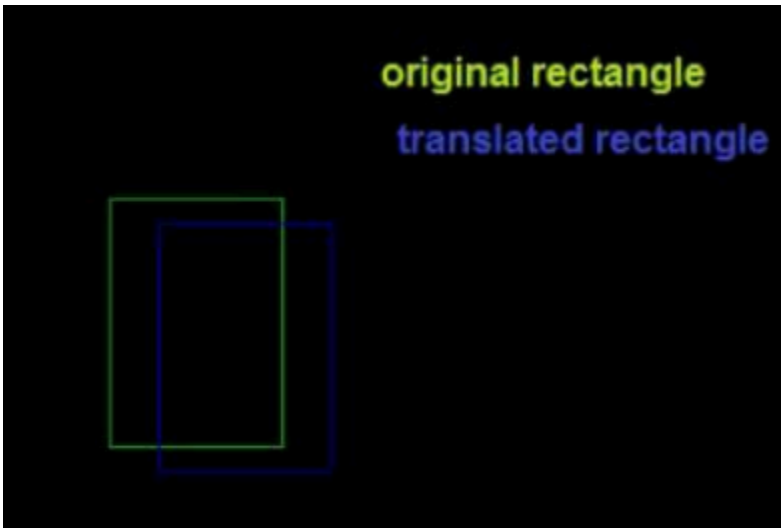
Example:



Program to demonstrate Translation:

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

void RectAngle(int x, int y, int Height, int Width);
void Translate(int x, int y, int Height, int Width);

void main() {
   int gd = DETECT, gm;
   int x, y, Height, Width;
   initgraph(&gd, &gm, " ");
   printf("Enter the First point for the Rectangle:");
   scanf("%d%d", &x, &y);
   printf("Enter the Height&Width for the Rectangle:");
   scanf("%d%d", &Height, &Width);
   RectAngle(x, y, Height, Width);
   getch();
   cleardevice();
   Translate(x, y, Height, Width);
   RectAngle(x, y, Height, Width);
   getch();
}

void RectAngle(int x, int y, int Height, int Width) {
   line(x, y, x + Width, y);
   line(x, y, x, y + Height);
   line(x + Width, y, x + Width, y + Height);
   line(x, y + Height, x + Width, y + Height);
}

void Translate(int x, int y, int Height, int Width) {
   int Newx, Newy, a, b;
   printf("Enter the Transaction coordinates");
   scanf("%d%d", &Newx, &Newy);
   cleardevice();
   a = x + Newx;
   b = y + Newy;

  RectAngle(a, b, Height, Width);
}
```

OUTPUT:

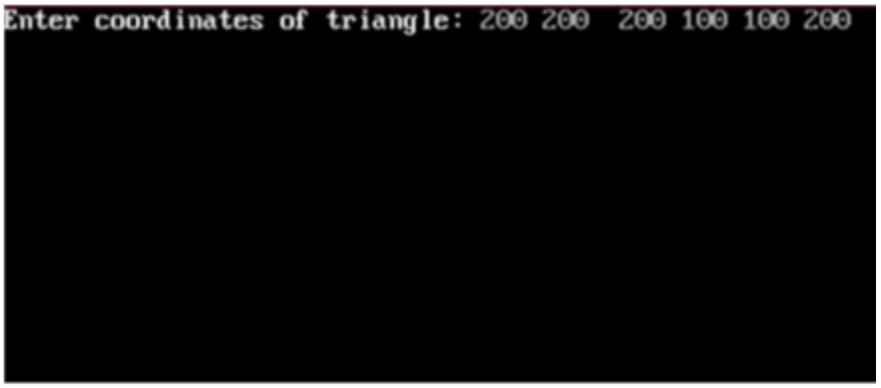


Program to demonstrate Rotation:

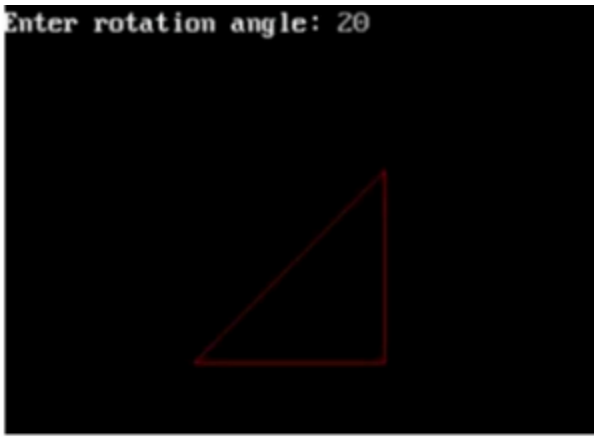
```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
main()
{
   intgd=0,gm,x1,y1,x2,y2,x3,y3;
   double s,c, angle;
   initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
   setcolor(RED);
   printf("Enter coordinates of triangle: ");
   scanf("%d%d%d%d%d%d",&x1,&y1,&x2,&y2, &x3, &y3);
   setbkcolor(WHITE);
   cleardevice();
   line(x1,y1,x2,y2);
   line(x2,y2, x3,y3);
   line(x3, y3, x1, y1);
   getch();
   setbkcolor(BLACK);
   printf("Enter rotation angle: ");
   scanf("%lf", &angle);
   setbkcolor(WHITE);
   c = cos(angle *M_PI/180);
   s = sin(angle *M_PI/180);
   x1 = floor(x1 * c + y1 * s);
   y1 = floor(-x1 * s + y1 * c);
   x2 = floor(x2 * c + y2 * s);
   y2 = floor(-x2 * s + y2 * c);
   x3 = floor(x3 * c + y3 * s);
   y3 = floor(-x3 * s + y3 * c);
   cleardevice();
   line(x1, y1 ,x2, y2);
   line(x2,y2, x3,y3);
   line(x3, y3, x1, y1);
   getch();
   closegraph();
   return 0;
}
```

OUTPUT:

Before Rotation:



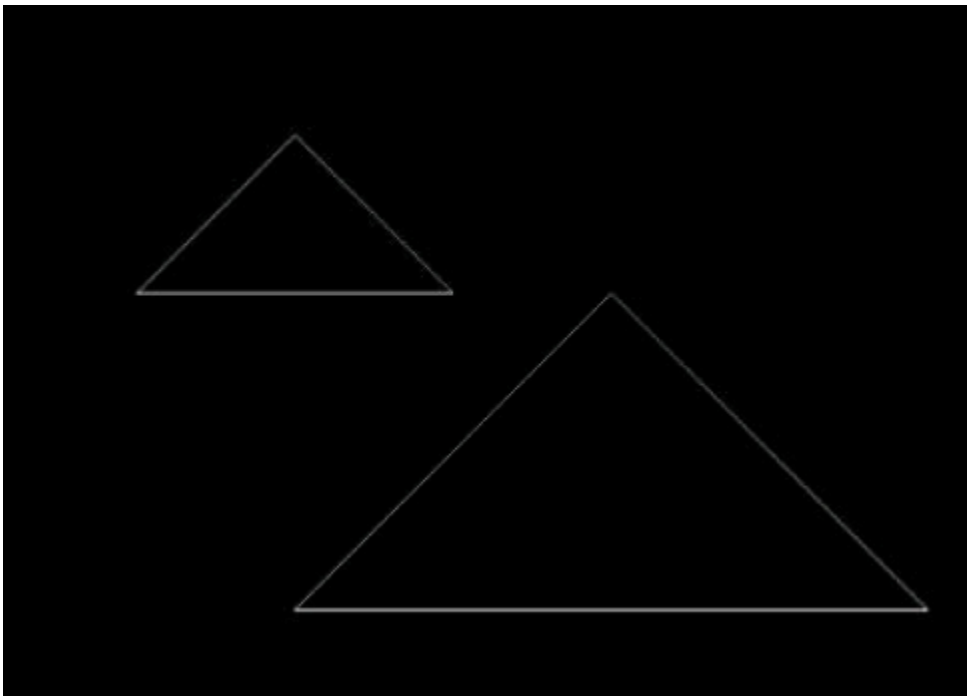
After Rotation:



Program to demonstrate Scaling:

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

void findNewCoordinate(int s[][2], int p[][1])
{
   int temp[2][1] = { 0 };

   for (int i = 0; i < 2; i++)
      for (int j = 0; j < 1; j++)
         for (int k = 0; k < 2; k++)
            temp[i][j] += (s[i][k] * p[k][j]);

   p[0][0] = temp[0][0];
   p[1][0] = temp[1][0];

}
void scale(int x[], int y[], int sx, int sy)
{
   line(x[0], y[0], x[1], y[1]);
   line(x[1], y[1], x[2], y[2]);
   line(x[2], y[2], x[0], y[0]);
   int s[2][2] = { sx, 0, 0, sy };
   int p[2][1];
   for (int i = 0; i < 3; i++)
   {
      p[0][0] = x[i];
      p[1][0] = y[i];

      findNewCoordinate(s, p);

      x[i] = p[0][0];
      y[i] = p[1][0];
   }
   line(x[0], y[0], x[1], y[1]);
   line(x[1], y[1], x[2], y[2]);
   line(x[2], y[2], x[0], y[0]);
}

int main()
{
   int x[] = { 100, 200, 300 };
   int y[] = { 200, 100, 200 };
   int sx = 2, sy = 2;

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

   scale(x, y, sx,sy);
   getch();

   return 0;
```

OUTPUT:

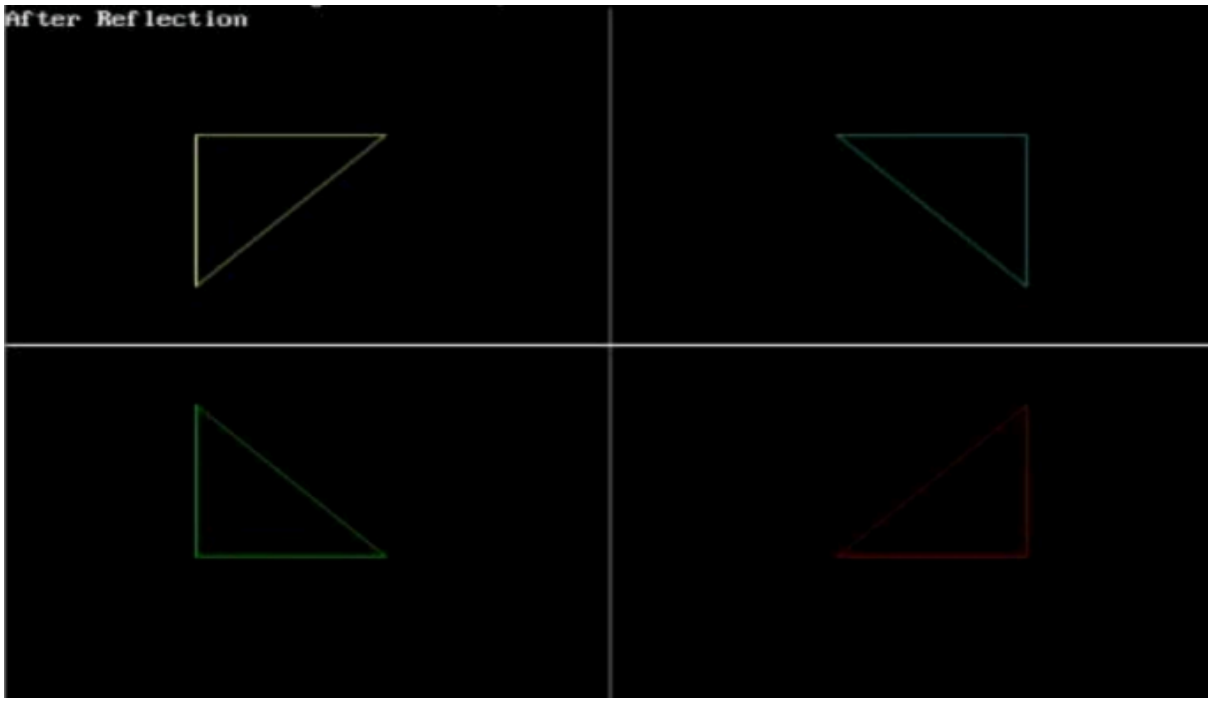


Program to demonstrate Reflection:

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

void main()
{
   int gm, gd = DETECT, ax, x1 = 100;
   int x2 = 100, x3 = 200, y1 = 100;
   int y2 = 200, y3 = 100;
   initgraph(&gd, &gm, "");
   cleardevice();
   line(getmaxx() / 2, 0, getmaxx() / 2,
       getmaxy());
   line(0, getmaxy() / 2, getmaxx(),
       getmaxy() / 2);
   printf("Before Reflection Object"
         " in 2nd Quadrant");
   setcolor(14);
   line(x1, y1, x2, y2);
   line(x2, y2, x3, y3);
   line(x3, y3, x1, y1);
   getch();
```

```
    printf("\nAfter Reflection");
    setcolor(4);
    line(getmaxx() - x1, getmaxy() - y1,
        getmaxx() - x2, getmaxy() - y2);

    line(getmaxx() - x2, getmaxy() - y2,
        getmaxx() - x3, getmaxy() - y3);

    line(getmaxx() - x3, getmaxy() - y3,
        getmaxx() - x1, getmaxy() - y1);
    setcolor(3);
    line(getmaxx() - x1, y1,
        getmaxx() - x2, y2);
    line(getmaxx() - x2, y2,
        getmaxx() - x3, y3);
    line(getmaxx() - x3, y3,
        getmaxx() - x1, y1);
    setcolor(2);
    line(x1, getmaxy() - y1, x2,
        getmaxy() - y2);
    line(x2, getmaxy() - y2, x3,
        getmaxy() - y3);
    line(x3, getmaxy() - y3, x1,
        getmaxy() - y1);
    getch();
    closegraph();
}
```

OUTPUT:



Program to demonstrate Shear:

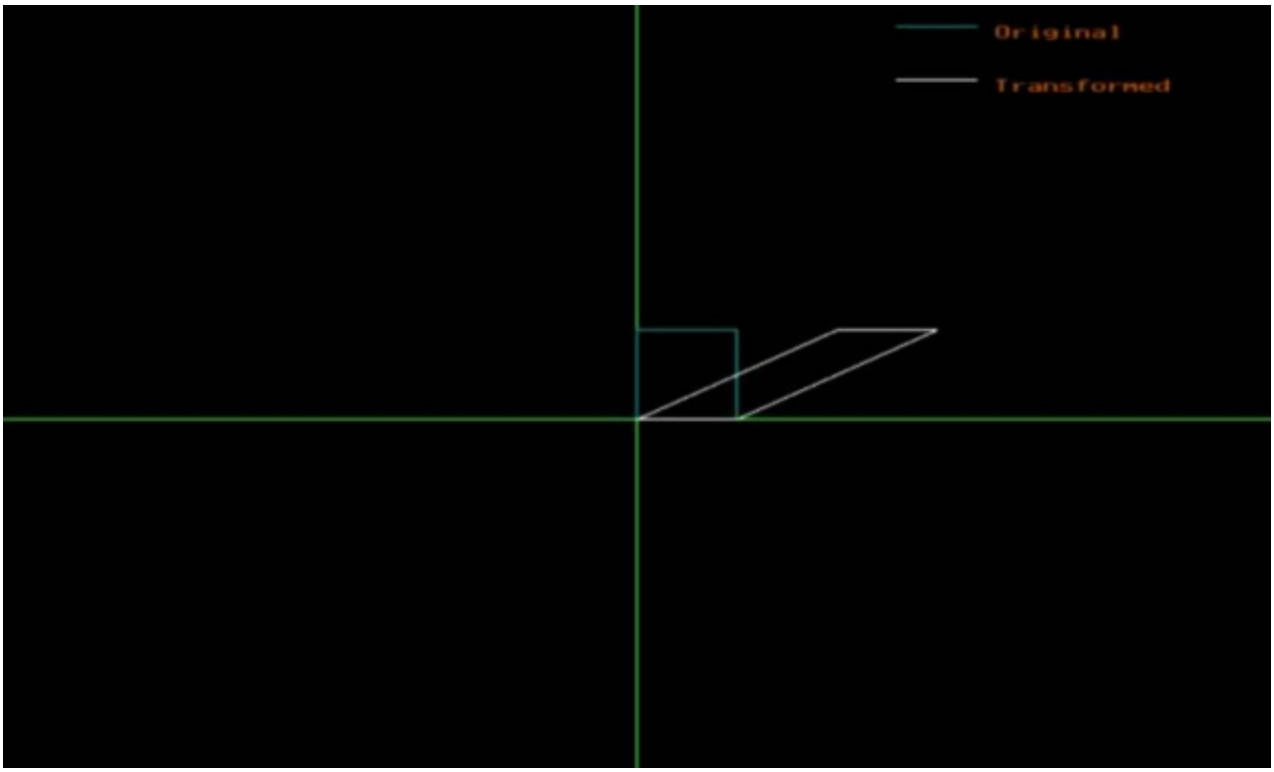
```
#include<iostream.h>
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<dos.h>

void mul(int mat[3][3],int vertex[10][3],int n);
void shear(int vertex[10][3],int n);

void init(int vertex[10][3],int n);
int main()
{
  int i,x,y;
  int vertex[10][3],n;
  clrscr();
  cout<<"\nEnter the no. of vertex : ";
  cin>>n;
   for(i=0;i<n;i++)
     {
            cout<<"Enter the points (x,y): ";
            cin>>x>>y;
            vertex[i][0]=x;
            vertex[i][1]=y;
            vertex[i][2]=1;
     }
     shear(vertex,n);

          getch();
          return 0;
}

void init(int vertex[10][3],int n)
{
   int gd=DETECT,gm,i;
   initgraph(&gd,&gm,"C:\\turboc3\\bgi");
   setcolor(10);
   line(0,240,640,240);
   line(320,0,320,480);
   setcolor(3);
   line(450,20,490,20);
```

```
   setcolor(15);
   line(450,50,490,50);
   setcolor(6);
   outtextxy(500,20,"Original");
   outtextxy(500,50,"Transformed");
   setcolor(3);

   for(i=0;i<n-1;i++)
    {
      line(320+vertex[i][0],240-vertex[i][1],320+vertex[i+1][0],240-vertex[i+1][1]);
    }
    line(320+vertex[n-1][0],240-vertex[n-1][1],320+vertex[0][0],240-vertex[0][1]);

}

void mul(int mat[3][3],int vertex[10][3],int n)
{
  int i,j,k;
 int res[10][3];
  for(i=0;i<n;i++)
   {
     for(j=0;j<3;j++)
      {
              res[i][j]=0;
        for(k=0;k<3;k++)
              {
                res[i][j] = res[i][j] + vertex[i][k]*mat[k][j];
               }
    }
   }
   setcolor(15);
   for(i=0;i<n-1;i++)
    {
      line(320+res[i][0],240-res[i][1],320+res[i+1][0],240-res[i+1][1]);
    }
    line(320+res[n-1][0],240-res[n-1][1],320+res[0][0],240-res[0][1]);

}

void shear(int vertex[10][3],int n)
```

```
{
  int opt;
  int shear_array[3][3];
  cout<<"\n1.x-shear\n2.y-shear\nYour Choice: ";
  cin>>opt;
  switch(opt)
  {
   case 1: int xsh;
                    cout<<"\nEnter the x shear : ";
             cin>>xsh;
                shear_array[0][0]=1;
           shear_array[1][0]=xsh;
           shear_array[2][0]=0;
           shear_array[0][1]=0;
           shear_array[1][1]=1;
           shear_array[2][1]=0;
           shear_array[0][2]=0;
           shear_array[1][2]=0;
           shear_array[2][2]=1;
           init(vertex,n);
           mul(shear_array,vertex,n);
            break;
       case 2:int ysh;
             cout<<"\nEnter the y shear : ";
             cin>>ysh;
                shear_array[0][0]=1;
          shear_array[1][0]=0;
          shear_array[2][0]=0;
          shear_array[0][1]=ysh;
          shear_array[1][1]=1;
          shear_array[2][1]=0;
          shear_array[0][2]=0;
          shear_array[1][2]=0;
          shear_array[2][2]=1;
           init(vertex,n);
           mul(shear_array,vertex,n);
            break;
    }
}
```

OUTPUT:

```
Enter the no. of vertex : 4
Enter the points (x,y): 0 0
Enter the points (x,y): 50 0
Enter the points (x,y): 50 50
Enter the points (x,y): 0 50

1.x-shear
2.y-shear
Your Choice: 1

Enter the x shear : 2_
```



Conclusion: We have successfully implemented 2D transformations in the form of translation, rotation, scaling, reflection & shear.