ARYAMAN MISHRA

19BCE1027

```
#include <stdio.h>
#include <graphics.h>
#include <math.h>
#include <stdlib.h>
#include <dos.h>
#include <conio.h>
#define ORG -50
# define  f               0.3
 # define  projection_angle   45
 void show_screen( );


 void apply_x_shearing(int[5][3],constfloat,constfloat);
 void multiply_matrices(constfloat[4],constfloat[4][4],float[4]);


 void draw_pyramid(constint [5][3]);
 void get_projected_point(int&,int&,int&);


 void Line(constint,constint,constint,constint);
  void show_screen( );


 void apply_y_shearing(int[5][3],constfloat,constfloat);
 void multiply_matrices(constfloat[4],constfloat[4][4],float[4]);

void trans();
#define ORG -50

double face1[5][2] = {
    { 250, 125 },
    { 350, 125 },
    { 350, 225 },
    { 250, 225 },
    { 250, 125 }
         };

double face2[5][2] = {
    { 250+ORG, 125-ORG },
    { 350+ORG, 125-ORG },
    { 350+ORG, 225-ORG },
    { 250+ORG, 225-ORG },
    { 250+ORG, 125-ORG }
         };

double angle = 5.0 * M_PI / 180;
double midx1, midy1, midx2, midy2;
```

```
void rotate (void)

{

   int i;

   for (i=0; i<5; i++)

   {

 double xnew, ynew;


 xnew = midx1 + (face1[i][0] - midx1) * cos (angle) -

      (face1[i][1] - midy1) * sin (angle);

 ynew = midy1 + (face1[i][0] - midx1) * sin (angle) +

  (face1[i][1] - midy1) * cos (angle);


 face1[i][0] = xnew;

 face1[i][1] = ynew;


 xnew = midx2 + (face2[i][0] - midx2) * cos (angle) -

  (face2[i][1] - midy2) * sin (angle);

 ynew = midy2 + (face2[i][0] - midx2) * sin (angle) +

  (face2[i][1] - midy2) * cos (angle);


 face2[i][0] = xnew;

 face2[i][1] = ynew;

   }
   cleardevice();


   for (i=0; i<4; i++)
   {
 setcolor(7);
 line (face1[i][0], face1[i][1], face1[i+1][0], face1[i+1][1]);
 setcolor(8);
 line (face2[i][0], face2[i][1], face2[i+1][0], face2[i+1][1]);
 setcolor(9);
 line (face1[i][0], face1[i][1], face2[i][0], face2[i][1]);
   }


   delay (125);
}
void apply_y_shearing(int edge_points[5][3],constfloat a,constfloat b)
   {
     for(int count=0;count<5;count++)
    {
       float matrix_a[4]={edge_points[count][0],edge_points[count][1],
                    edge_points[count][2],1};
       float matrix_b[4][4]={
               { 1,0,0,0 } ,
               { a,1,b,0 } ,
               { 0,0,1,0 } ,
                 { 0,0,0,1 }
                };

        float matrix_c[4]={0};

        multiply_matrices(matrix_a,matrix_b,matrix_c);

        edge_points[count][0]=(int)(matrix_c[0]+0.5);
        edge_points[count][1]=(int)(matrix_c[1]+0.5);
        edge_points[count][2]=(int)(matrix_c[2]+0.5);
      }
   }

//these are left,top,right,bottom parameters for bar3d function
int maxx,maxy,midx,midy;
//function for translation of a 3d object
void trans()

{

         int x,y,z,o,x1,x2,y1,y2;

         midx=200;
```

```
          midy=200;


          //function to draw 3D rectangular box

          bar3d(midx+50,midy-100,midx+100,midy-50,20,1);

 delay(1000);

          printf("Enter translation factor");


          scanf("%d%d",&x,&y);


          printf("After translation:");


          bar3d(midx+x+50,midy-(y+100),midx+x+100,midy-(y+50),20,1);


}

int x1,x2,y1,y2,mx,my,depth;

void draw();

void rotate();

void rotate()

{

   float t;

   int a1,b1,a2,b2,dep;

   printf("Enter the angle to rotate=");

   scanf("%f",&t);

   t=t*(3.14/180);
```

```
    a1=mx+(x1-mx)*cos(t)-(y1-my)*sin(t);

    a2=mx+(x2-mx)*cos(t)-(y2-my)*sin(t);

    b1=my+(x1-mx)*sin(t)-(y1-my)*cos(t);

    b2=my+(x2-mx)*sin(t)-(y2-my)*cos(t);

    if(a2>a1)

       dep=(a2-a1)/4;

    else

      dep=(a1-a2)/4;

    bar3d(a1,b1,a2,b2,dep,1);

    setcolor(5);

    //draw();


}

 void draw()

{

    bar3d(x1,y1,x2,y2,depth,1);

}

void scale();

//these are left,top,right,bottom parameters for bar3d function

int maxx,maxy,midx,midy;

//function for scaling of a 3d object

void scale()

{

            int x,y,z,o,x1,x2,y1,y2;
```

```
        midx=200;

        midy=200;

        bar3d(midx+50,midy-100,midx+100,midy-50,20,0);

        printf("before scaling\n");

        printf("Enter scaling factors\n");

        scanf("%d %d %d", &x,&y,&z);

        printf("After scaling\n");

        bar3d(midx+(x*50),midy-(y*100),midx+(x*100),midy-(y*50),20*z,1);

}


 void Line(constint,constint,constint,constint);

double face1[5][2] = {

    { 250, 125 },

    { 350, 125 },

    { 350, 225 },

    { 250, 225 },

    { 250, 125 }

        };


double face2[5][2] = {

    { 250+ORG, 125-ORG },

    { 350+ORG, 125-ORG },

    { 350+ORG, 225-ORG },

    { 250+ORG, 225-ORG },
      { 250+ORG, 125-ORG }
            };

double angle = 5.0 * M_PI / 180;
double midx1, midy1, midx2, midy2;

void rotate (void)
{
   int i;
   for (i=0; i<5; i++)
   {
 double xnew, ynew;

 xnew = midx1 + (face1[i][0] - midx1) * cos (angle) -
        (face1[i][1] - midy1) * sin (angle);
 ynew = midy1 + (face1[i][0] - midx1) * sin (angle) +
  (face1[i][1] - midy1) * cos (angle);

 face1[i][0] = xnew;
 face1[i][1] = ynew;

 xnew = midx2 + (face2[i][0] - midx2) * cos (angle) -
  (face2[i][1] - midy2) * sin (angle);
 ynew = midy2 + (face2[i][0] - midx2) * sin (angle) +
  (face2[i][1] - midy2) * cos (angle);


 face2[i][0] = xnew;

 face2[i][1] = ynew;

   }


   cleardevice();


   for (i=0; i<4; i++)

   {

 setcolor(7);

 line (face1[i][0], face1[i][1], face1[i+1][0], face1[i+1][1]);

 setcolor(8);

 line (face2[i][0], face2[i][1], face2[i+1][0], face2[i+1][1]);

 setcolor(9);

 line (face1[i][0], face1[i][1], face2[i][0], face2[i][1]);

   }


   delay (125);

}



 void show_screen( );
 void apply_x_shearing(int[5][3],constfloat,constfloat);
 void multiply_matrices(constfloat[4],constfloat[4][4],float[4]);

 void draw_pyramid(constint [5][3]);
 void get_projected_point(int&,int&,int&);

 void Line(constint,constint,constint,constint);


 int main( )
   {
      int driver=VGA;
      int mode=VGAHI;

      initgraph(&driver,&mode,"C:\\TURBOC3\\BGI");

      show_screen( );

      int pyramid[5][3]={
           {280,220,40},     //  base front left
           {360,220,40},     //  base front right
           {360,220,-40},    //  base back right
           {280,220,-40},    //  base back left
           {320,100,0}       //  top
```

```
       };

    setcolor(15);
   draw_pyramid(pyramid);

    setcolor(15);
    settextstyle(0,0,1);
   outtextxy(50,415,"*** Press any key to see the 3D Shearing along x-axis.");

    apply_x_shearing(pyramid,0.4,0.3);

    getch( );

    setcolor(10);
   draw_pyramid(pyramid);

    getch( );
    return 0;
  }

void apply_x_shearing(int edge_points[5][3],constfloat a,constfloat b)
  {
    for(int count=0;count<5;count++)
   {
     float matrix_a[4]={edge_points[count][0],edge_points[count][1],
               edge_points[count][2],1};
     float matrix_b[4][4]={
          { 1,a,b,0 } ,
          { 0,1,0,0 } ,
          { 0,0,1,0 } ,
          { 0,0,0,1 }
         };

     float matrix_c[4]={0};

     multiply_matrices(matrix_a,matrix_b,matrix_c);

     edge_points[count][0]=(int)(matrix_c[0]+0.5);
     edge_points[count][1]=(int)(matrix_c[1]+0.5);
     edge_points[count][2]=(int)(matrix_c[2]+0.5);
   }
  }

void multiply_matrices(constfloat matrix_1[4],
         constfloat matrix_2[4][4],float matrix_3[4])
  {
    for(int count_1=0;count_1<4;count_1++)
```

```
      {
         for(int count_2=0;count_2<4;count_2++)
        matrix_3[count_1]+=
              (matrix_1[count_2]*matrix_2[count_2][count_1]);
      }
    }

void draw_pyramid(constint points[5][3])
    {
       int edge_points[5][3];

       for(int i=0;i<5;i++)
      {
         edge_points[i][0]=points[i][0];
         edge_points[i][1]=points[i][1];
         edge_points[i][2]=points[i][2];

         get_projected_point(edge_points[i][0],
                   edge_points[i][1],edge_points[i][2]);
      }

       Line(edge_points[0][0],edge_points[0][1],
                     edge_points[1][0],edge_points[1][1]);
       Line(edge_points[1][0],edge_points[1][1],
               edge_points[2][0],edge_points[2][1]);
     Line(edge_points[2][0],edge_points[2][1],
               edge_points[3][0],edge_points[3][1]);
     Line(edge_points[3][0],edge_points[3][1],
               edge_points[0][0],edge_points[0][1]);

     Line(edge_points[0][0],edge_points[0][1],
               edge_points[4][0],edge_points[4][1]);
     Line(edge_points[1][0],edge_points[1][1],
               edge_points[4][0],edge_points[4][1]);
     Line(edge_points[2][0],edge_points[2][1],
               edge_points[4][0],edge_points[4][1]);
     Line(edge_points[3][0],edge_points[3][1],
               edge_points[4][0],edge_points[4][1]);
   }

void get_projected_point(int& x,int& y,int& z)
   {
     float fcos0=(f*cos(projection_angle*(M_PI/180)));
     float fsin0=(f*sin(projection_angle*(M_PI/180)));

     float Par_v[4][4]={
           {1,0,0,0},
           {0,1,0,0},
           {fcos0,fsin0,0,0},

           {0,0,0,1}

         };


     float xy[4]={x,y,z,1};

     float new_xy[4]={0};


     multiply_matrices(xy,Par_v,new_xy);


     x=(int)(new_xy[0]+0.5);

     y=(int)(new_xy[1]+0.5);

     z=(int)(new_xy[2]+0.5);

   }


void Line(constint x_1,constint y_1,constint x_2,constint y_2)

   {

     int color=getcolor( );


     int x1=x_1;

     int y1=y_1;


     int x2=x_2;

     int y2=y_2;
```

```
 if(x_1>x_2)
{
   x1=x_2;
   y1=y_2;

   x2=x_1;
   y2=y_1;
}

 int dx=abs(x2-x1);
 int dy=abs(y2-y1);
 int inc_dec=((y2>=y1)?1:-1);

 if(dx>dy)
{
   int two_dy=(2*dy);
   int two_dy_dx=(2*(dy-dx));
   int p=((2*dy)-dx);

   int x=x1;
   int y=y1;

   putpixel(x,y,color);
   while(x<x2)
  {
     x++;

     if(p<0)
        p+=two_dy;

     else
        {
       y+=inc_dec;
       p+=two_dy_dx;
        }

     putpixel(x,y,color);
  }
}

 else
{
   int two_dx=(2*dx);
   int two_dx_dy=(2*(dx-dy));
   int p=((2*dx)-dy);

   int x=x1;
```

```
        int y=y1;

        putpixel(x,y,color);

        while(y!=y2)
       {
          y+=inc_dec;

          if(p<0)
             p+=two_dx;

          else
             {
            x++;
            p+=two_dx_dy;
             }

          putpixel(x,y,color);
       }
     }
   }

void show_screen( )
   {
      setfillstyle(1,1);

    bar(210,26,420,38);


      settextstyle(0,0,1);

    setcolor(15);


outtextxy(5,5,"******************************************************************************************");

      outtextxy(5,17,"*-**************************************************************************************-*");

      outtextxy(5,29,"*---------------------------                              --------------------------*");

      outtextxy(5,41,"*-**************************************************************************************-*");

      outtextxy(5,53,"*-**************************************************************************************-*");


    setcolor(11);

      outtextxy(218,29,"3D Shearing along x-axis");


    setcolor(15);


      for(int count=0;count<=30;count++)

         outtextxy(5,(65+(count*12)),"*-*                                                                                *-*");


      outtextxy(5,438,"*-**************************************************************************************-*");
      outtextxy(5,450,"*------------------------                        ------------------------*");

outtextxy(5,462,"****************************************************************************");


    setcolor(12);

      outtextxy(229,450,"Press any Key to exit.");

   }

int main () {

 char choice;

 printf("Enter 1 for translation,2 for reflection,3 for rotation,4 for scaling,5 for shearing along x axis,6 for shearing along y axis.\n");

 scanf("%c", &choice)

 switch(choice) {

 case '1' :int ch;


           int gd=DETECT,gm;


           detectgraph(&gd,&gm);


           initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");


           trans();

break;

case '2' :
 int gd = DETECT, gm;

   midx1 = (face1[0][0] + face1[1][0]) / 2.0;
   midy1 = (face1[1][1] + face1[2][1]) / 2.0;
   midx2 = (face2[0][0] + face2[1][0]) / 2.0;
   midy2 = (face2[1][1] + face2[2][1]) / 2.0;

   initgraph (&gd, &gm, "C:\\TURBOC3\\BGI");

   while (!kbhit())
 rotate();

   closegraph();
break;
 case '3' :
int gd=DETECT,gm,c;
   initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
   printf("\n3D Transformation Rotating\n\n");
   printf("\nEnter 1st top value(x1,y1):");
   scanf("%d%d",&x1,&y1);
   printf("Enter right bottom value(x2,y2):");
   scanf("%d%d",&x2,&y2);
   depth=(x2-x1)/4;
   mx=(x1+x2)/2;
```

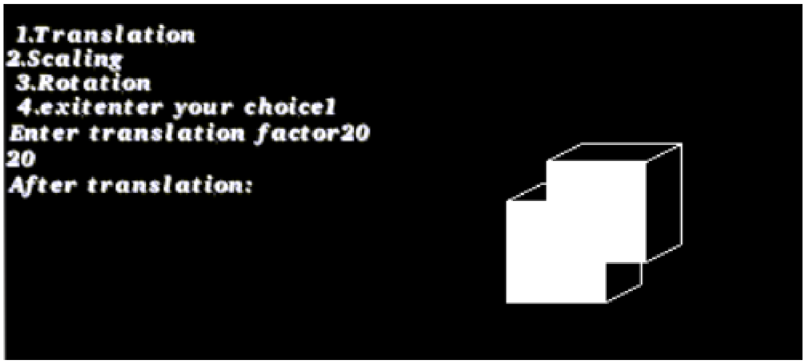
```
  my=(y1+y2)/2;
  draw();
  getch();
  cleardevice();
  rotate();
  getch();
break;
case '4' : int ch;
       int gd=DETECT,gm;
       detectgraph(&gd,&gm);
       initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
       scale();
break;
case '5':int driver=VGA;
    int mode=VGAHI;

    initgraph(&driver,&mode,"C:\\TURBOC3\\BGI");

    show_screen( );

    int pyramid[5][3]={
          {280,220,40},     //  base front left
          {360,220,40},     //  base front right
          {360,220,-40},    //  base back right
```

```
               {280,220,-40},     //  base back left
               {320,100,0}       //  top
            };

      setcolor(15);
     draw_pyramid(pyramid);

       setcolor(15);
       settextstyle(0,0,1);
     outtextxy(50,415,"*** Press any key to see the 3D Shearing along x-axis.");

       apply_x_shearing(pyramid,0.4,0.3);

       getch( );

       setcolor(10);
     draw_pyramid(pyramid);

       getch( );
break;
 case '6':int driver=VGA;
       int mode=VGAHI;

       initgraph(&driver,&mode,"C:\\TURBOC3\\BGI");
```

```
  show_screen( );

  int pyramid[5][3]={
         {270,300,50},      //  base front left
         {370,300,50},      //  base front right
         {370,300,-50},     //  base back right
         {270,300,-50},     //  base back left
         {320,150,0}       //  top
      };

 setcolor(15);
draw_pyramid(pyramid);

  setcolor(15);
  settextstyle(0,0,1);
outtextxy(50,415,"*** Press any key to see the 3D Shearing along y-axis.");

  apply_y_shearing(pyramid,0.5,0.1);

  getch( );

  setcolor(10);
draw_pyramid(pyramid);
```

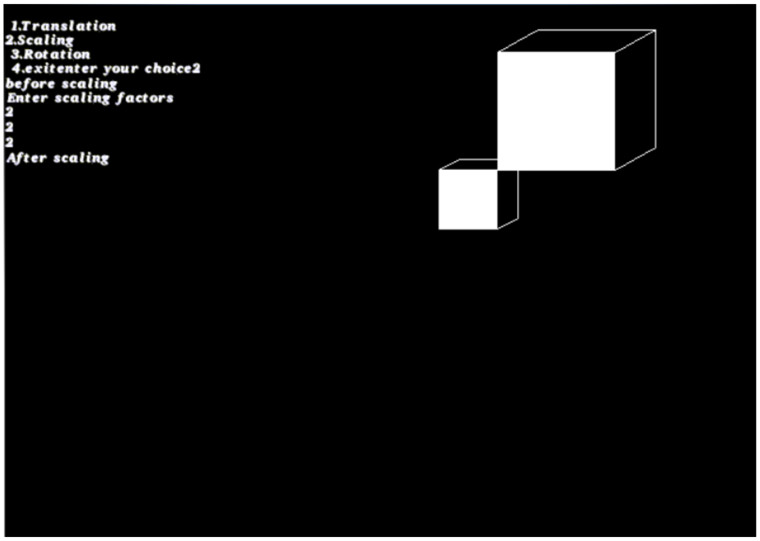
```
      getch( );
 default :
 printf("Wrong Choice.Try Again.\n" );
 }
 return 0;
}
```

TRANSLATION:



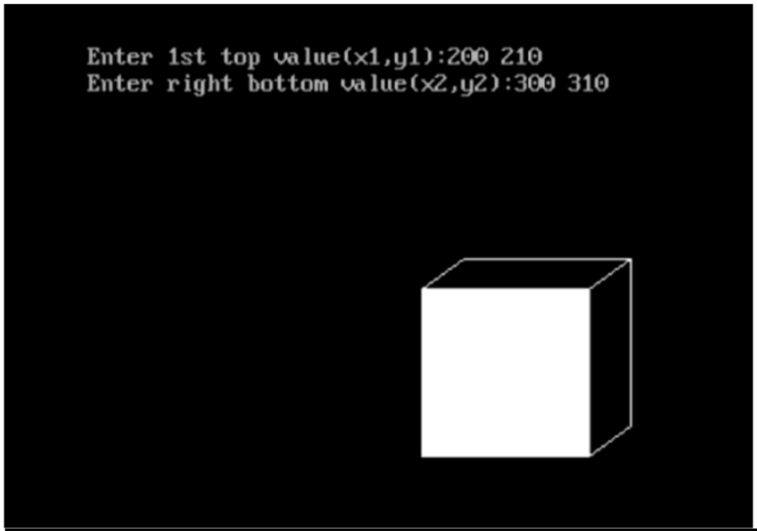
```
 1.Translation
2.Scaling
 3.Rotation
 4.exitenter your choice1
Enter translation factor20
20
After translation:
```

SCALING:

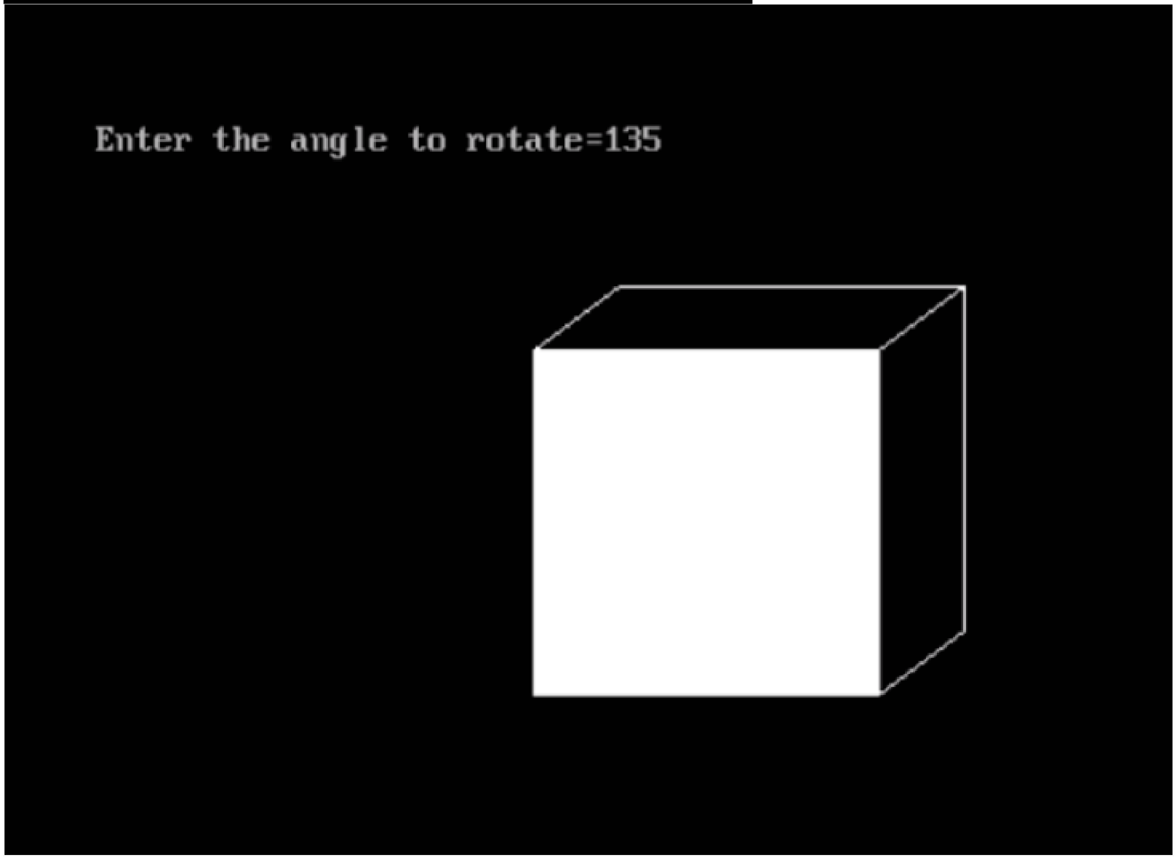


```
 1.Translation
2.Scaling
 3.Rotation
 4.exitenter your choice2
before scaling
Enter scaling factors
2
2
2
After scaling
```

ROTATION:

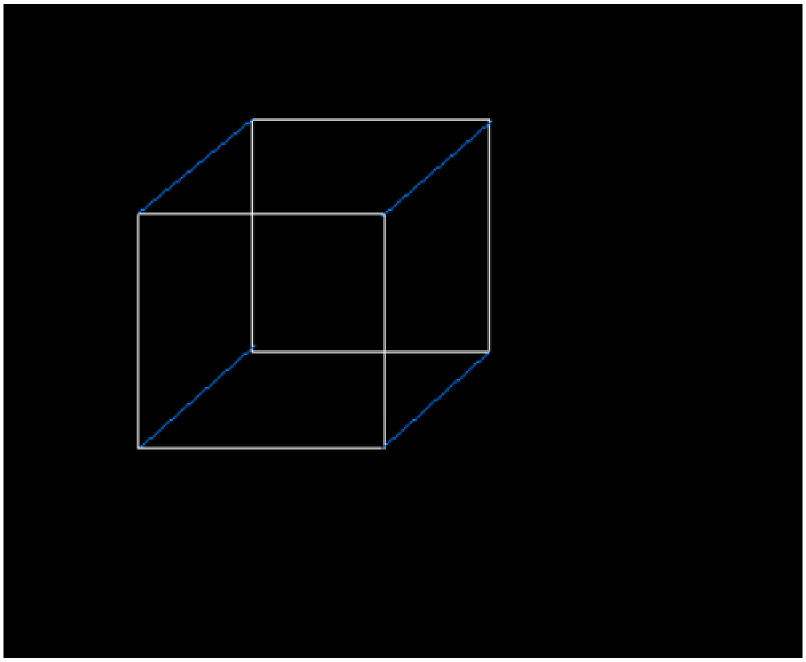


```
Enter 1st top value(x1,y1):200 210
Enter right bottom value(x2,y2):300 310
```



```
Enter the angle to rotate=135
```

REFLECTION:



SHEARING: