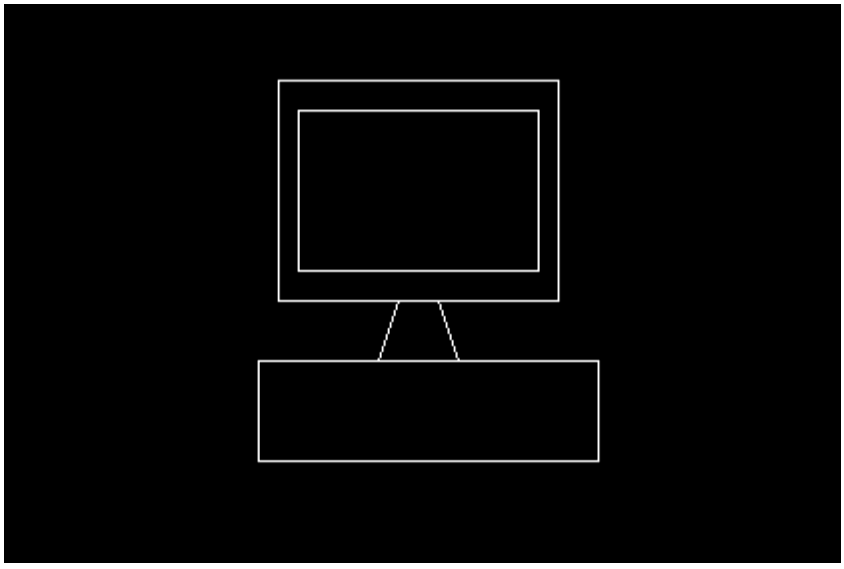
Practical no: 1

A)

```
 #include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
            {
               int gd,gm;
               detectgraph(&gd,&gm);
               initgraph(&gd,&gm,"c:\\turboc3\\bgi");
               rectangle(40,40,180,150);
               rectangle(50,55,170,135);
               line(100,150,90,180);
               line(120,150,130,180);
               rectangle(30,180,200,230);
              getch();
              closegraph();
               }
```

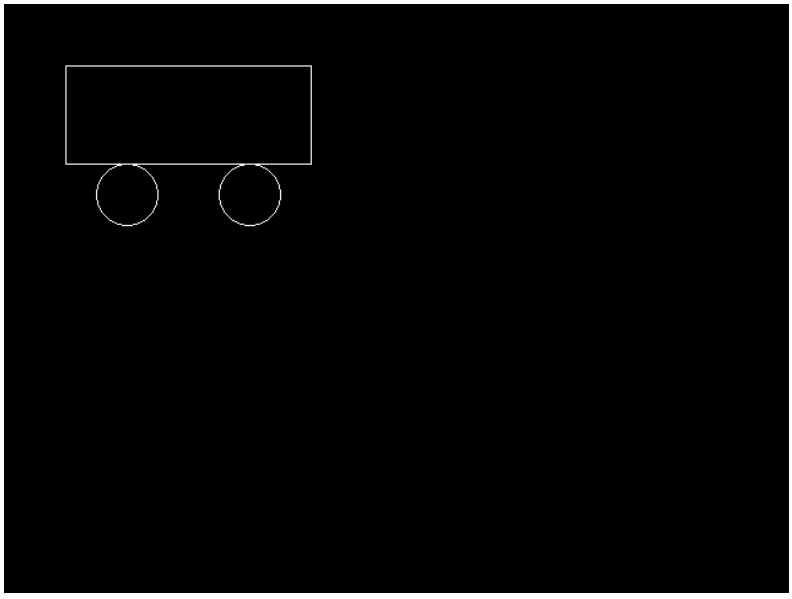
Output:



Z

B)

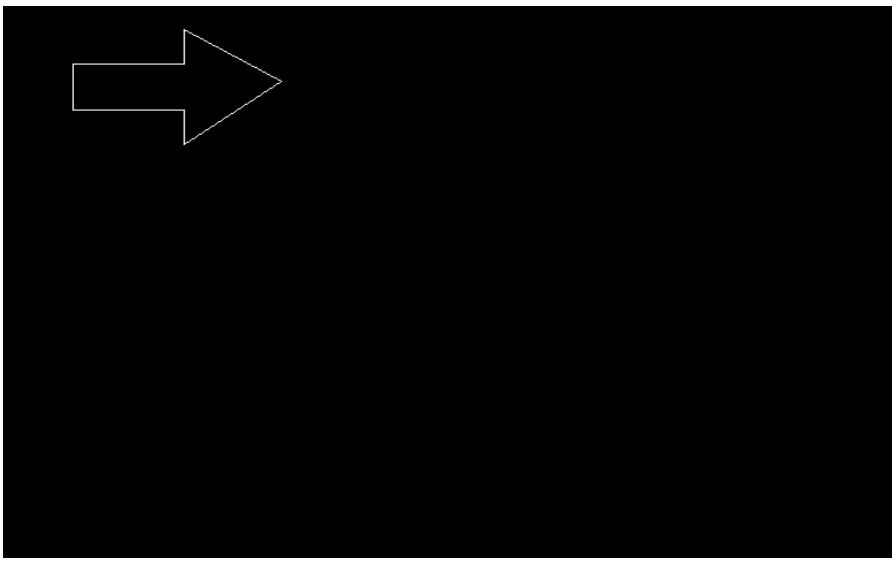
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
                    {
                     int gd,gm;
                      detectgraph(&gd,&gm);
                      intigraph(&gd,&gm,"c:\\turboc3\\bgi");
                       rectangle(50,50,250,130);
                      circle(100,155,25);
                      circle(200,155,25);
                      getch();
                       closegraph();
                  }
```



Z

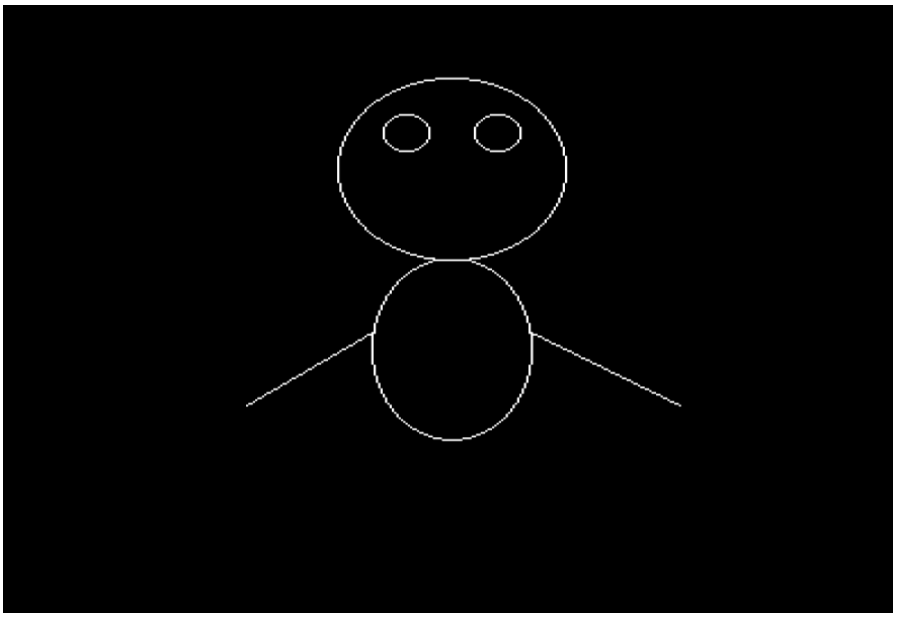
C)

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
{
int gd,gm;
detectgraph(&gd,&gm);
intigraph(&gd,&gm,"c:\\turboc3\\bgi");
line(50,50,50,90);
line(50,50,130,50);
line(50,90,130,90);
line(130,50,130,20);
line(130,90,130,120);
line(130,20,200,65);
line(200,65,130,120);
getch();
closegraph();
}
```



D)

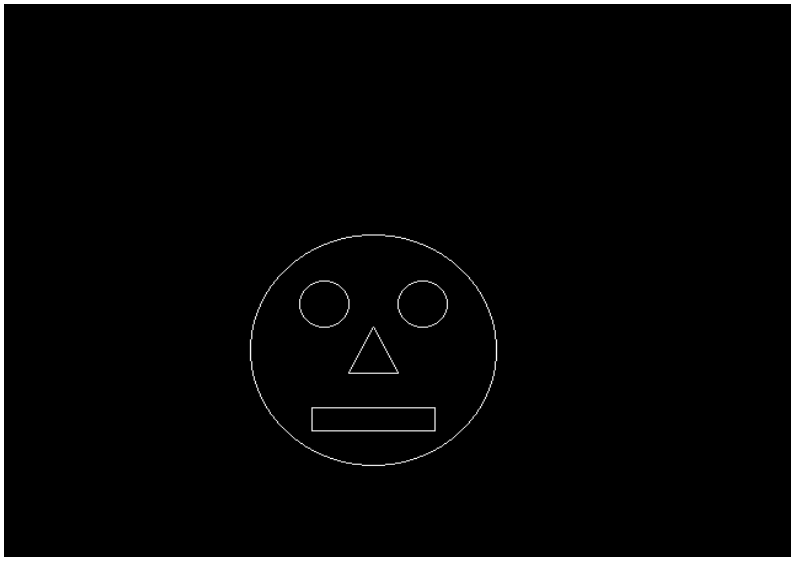
```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
        {
           int gd,gm;
           detectgraph(&gd,&gm);
           initgraph(&gd,&gm,"c:\\turboc3\\bgi");
           circle(90,90,50);
           circle(70,70,10);
           circle(110,70,10);
           ellipse(90,189,100,80,35,50);
           line(125,180,190,220);
           line(55,180,-250,400);

          getch();
          closegraph();
           }
```



Z

E)

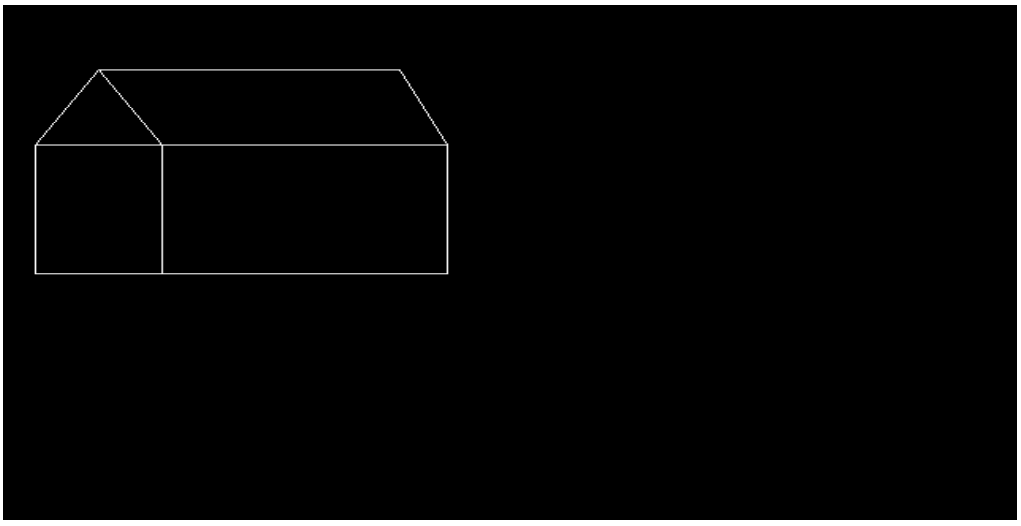
```
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
                        {
                          int gd,gm;
                          detectgraph(&gd,&gm);
                            initgraph(&gd,&gm,"c:\\turboc3\\bgi");
                           circle(300,300,100);
                           circle(260,260,20);
                           circle(340,260,20);
                           line(300,280,320,320);
                            line(320,320,280,320);
                           line(300,280,280,320);
                          rectangle(250,350,350,370);
                           getch();
                           closegraph();
                       }
```



Z

F)

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<dos.h>
void main()
                    {
                          int gd,gm;
                          detectgraph(&gd,&gm);

initgraph(&gd,&gm,"c:\\turboc3\\bgi");
                           line(60,60,100,130);
                           line(60,60,20,130);
                           line(20,130,100,130);
                           rectangle(20,130,100,250);
                          line(60,60,250,60);
                          line(250,60,280,130);
                            rectangle(100,130,280,250);
                         getch();
closegraph();
}
```



Z

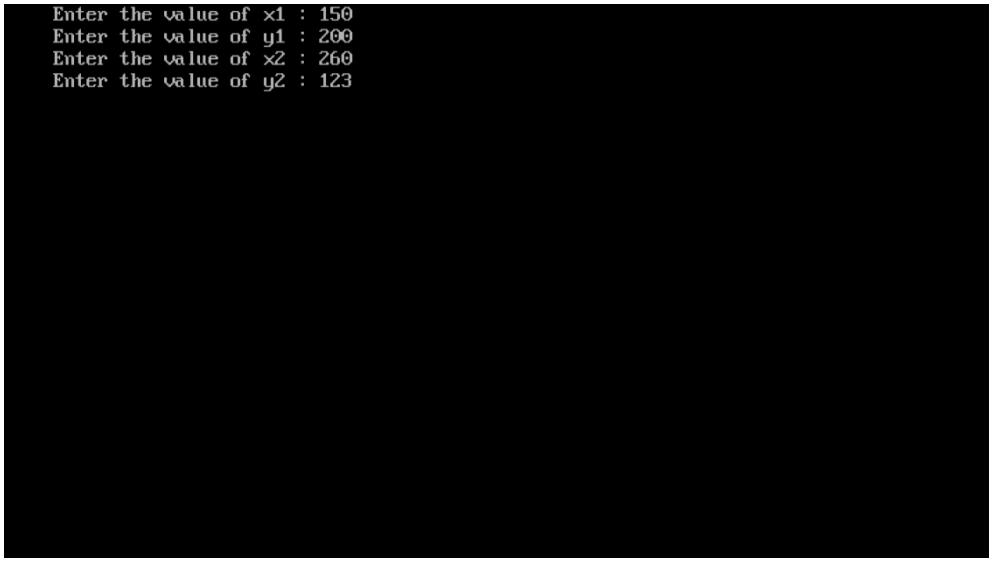
Practical no: 2

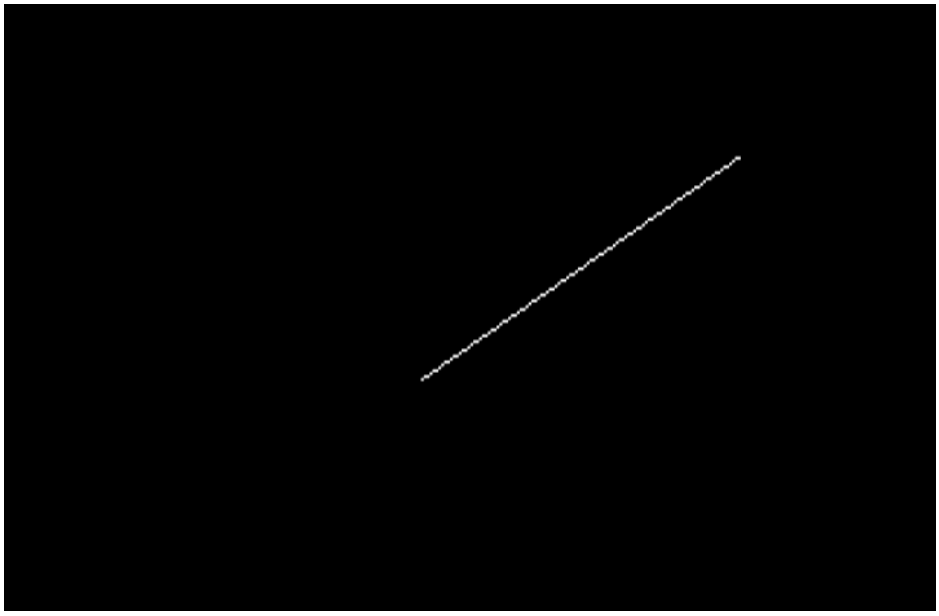
Program: -

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<conio.h>
#include<dos.h>
void main()
{
            float x,y,x1,y1,x2,y2,dx,dy,length;
            int I,gd=DETECT,gm;
            clrscr();
  /* Read two end points of line
---------------------------------- */
             printf("Enter the value of x1 :\t");
             scanf("%f",&x1);
             printf("Enter the value of y1 :\t");
             scanf("%f",&y1);
             printf("Enter the value of x2 :\t");
             scanf("%f",&x2);
             printf("Enter the value of y2 :\t");
             scanf("%f",&y2);
/* Initialise graphics mode
---------------------------------- */
//detectgraph(&gd,&gm);
            initgraph(&gd,&gm,"C:\\turboc3\\BGI");
            dx=abs(x2-x1);
            dy=abs(y2-y1);
            if (dx >= dy)
             {
              length = dx;
              }
             else
               {
              length = dy;
                 }
             dx = (x2-x1)/length;
             dy = (y2-y1)/length;
             x = x1 + 0.5;  /* Factor 0.5 is added to round
the values */

             y = y1 + 0.5;  /* Factor 0.5 is added to round
the values */
```

Z

```
            I = 1;          /* Initialise loop counter */
            while(I <= length)
             {
               putpixel(x,y,15);
               x = x + dx;
               y = y + dy;
               I = I + 1;
      delay(100); /* Delay is purposely inserted to see
                observe the line drawing process */
    }
        getch();
        closegraph();

}
```

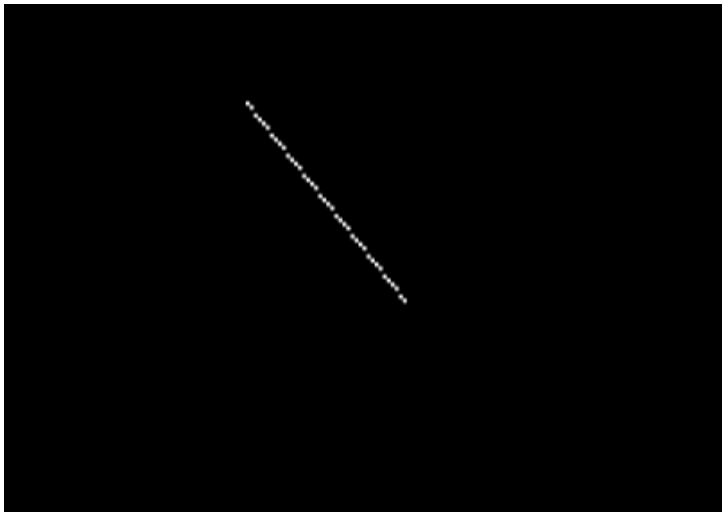




```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
#include<conio.h>
 void main()
{
          float x,y,x1,y1,x2,y2,dx,dy,e;
           int I,gd=DETECT,gm;
          clrscr();

/* Read two end points of line
---------------------------------- */
         printf("Enter the value of x1 :\t");
         scanf("%f",&x1);
         printf("Enter the value of y1 :\t");
          scanf("%f",&y1);
          printf("Enter the value of x2 :\t");
          scanf("%f",&x2);
          printf("Enter the value of y2 :\t");
          scanf("%f",&y2);

/* Initialise graphics mode
---------------------------------- */
//detectgraph(&gd,&gm);
         initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");


         dx=abs(x2-x1);
          dy=abs(y2-y1);

/* Initialise starting point
-----------------------------*/
          x = x1;
          y = y1;

/* Initialise decision variable
------------------------------- */

           e = 2 * dy-dx;

            I = 1;    /* Initialise loop counter */

do
{
```

Z

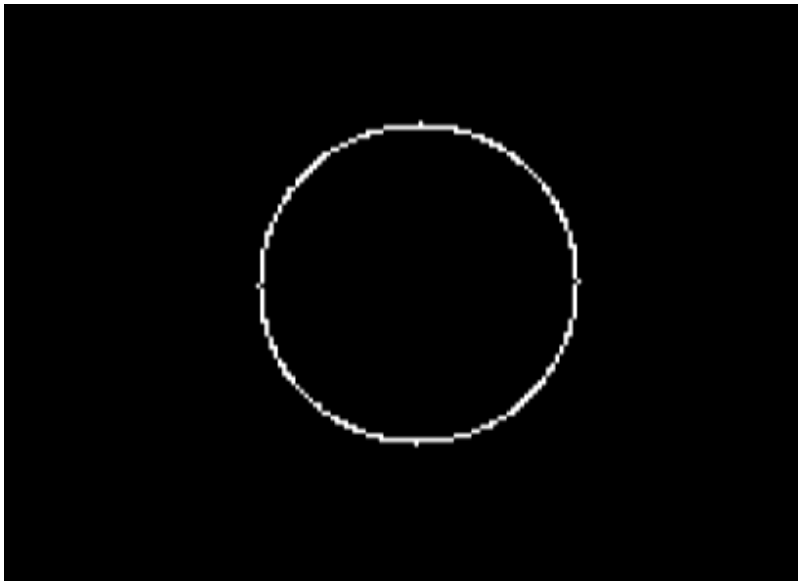
```
putpixel(x,y,15);
while(e >= 0)
    {
     y = y + 1;
     e = e - 2 * dx;
    }
x = x + 1;
e = e + 2 * dy;
I = I + 1;
 delay(100); /* Delay is purposely inserted to see
              observe the line drawing process */
}
while( I <= dx);
getch();
closegraph();

}
```



Practical No: 4

Circle using DDA Algorithm

```
#include<dos.h>
#include<conio.h>
#include<stdio.h>
#include<graphics.h>
#include<math.h>
void main()
           {
              float x1,y1,x2,y2,startx,starty,epsilon;
              int gd=DETECT,gm,I,val;
              int r;
              clrscr();
              printf("Enter the radius of a circle :");
              scanf("%d",&r);
/* Initialise graphics mode
---------------------------------- */
               initgraph(&gd,&gm,"c:\\turboc3\\bgi");
/* Initialise starting point
---------------------------------- */
           x1=r*cos(0);
           y1=r*sin(0);
           startx = x1;
           starty = y1;
  /*Calculations for epsilon
-----------------------------*/
i=0;
do
{
val = pow(2,i);
i++;
}while(val<r);
epsilon = 1/pow(2,i-1);
do
   {
    x2= x1 + y1*epsilon;
    y2 = y1 - epsilon*x2;
    putpixel(200+x2,200+y2,15);
 /* Reinitialise the current point
---------------------------------- */
    x1=x2;
    y1=y2;
        delay(100);   /* Delay is purposely inserted to see
                                                observe the line drawing process */
    }
  while( (y1 - starty ) < epsilon || (startx - x1) > epsilon);
getch();
closegraph();
}
```

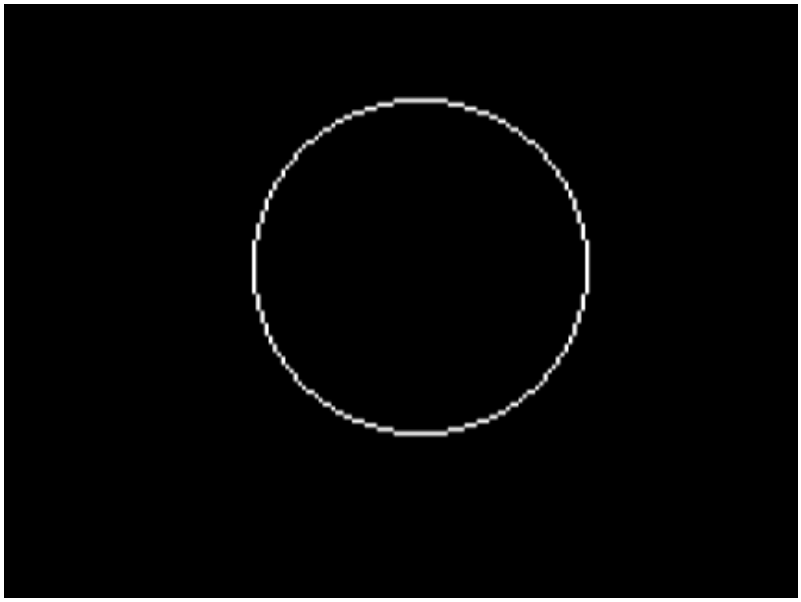


Z

Circle using the Bresenham's Algorithm
Program: -

```
#include<conio.h>
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<dos.h>
void main()
{
float d;
int gd=DETECT,gm,x,y;
int r;
clrscr();
/* Read the radius of the circle
---------------------------------- */
printf("Enter the radius of a circle :");
scanf("%d",&r);
/* Initialise graphics mode
------------------------------*/
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
/* Initialise starting points
------------------------------*/
x = 0;
y = r;
/* initialise the decision variable
--------------------------------------*/
d = 3 - 2 * r;
do
{      putpixel(200+x,200+y,15);
       putpixel(200+y,200+x,15);
       putpixel(200+y,200-x,15);
       putpixel(200+x,200-y,15);
       putpixel(200-x,200-y,15);
       putpixel(200-y,200-x,15);
       putpixel(200-y,200+x,15);
       putpixel(200-x,200+y,15);
       if (d <= 0)
       {
       d = d + 4*x + 6;
       }
       else
       {
       d = d + 4*(x-y) + 10;
       y = y - 1;
       }
       x = x + 1;
       delay(50);  /* Delay is purposely inserted to see
                      observe the line drawing process */
       }
       while(x < y);
getch();
closegraph();
}
```

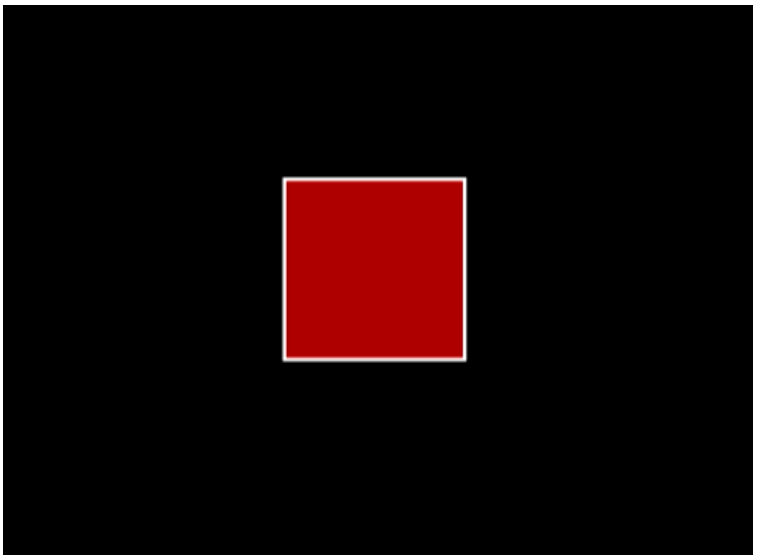
```
Enter the radius of a circle :40
```



Practical no: 5

```
/* Implement FloodFill Algorithm for polygon
filling(8connected region)
 */
#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <dos.h>
void main()
{
  int gd = DETECT, gm;
  void flood(int, int, int, int);
  /* Initialise graphics mode
  ---------------------------------- */

  initgraph(&gd, &gm, "c:\\turboc3\\bgi");
  rectangle(50, 50, 100, 100);
  flood(55, 55, 4, 15);
  getch();
  closegraph();
}

void flood(seed_x, seed_y, foreground_col, background_col)
{
  if (getpixel(seed_x, seed_y) != background_col &&
getpixel(seed_x, seed_y) != foreground_col)
  {
    delay(2);
    putpixel(seed_x, seed_y, foreground_col);
    flood(seed_x + 1, seed_y, foreground_col, background_col);
    flood(seed_x - 1, seed_y, foreground_col, background_col);
    flood(seed_x, seed_y + 1, foreground_col, background_col);
    flood(seed_x, seed_y - 1, foreground_col, background_col);
    flood(seed_x + 1, seed_y + 1, foreground_col,
background_col);
    flood(seed_x - 1, seed_y - 1, foreground_col,
background_col);
    flood(seed_x + 1, seed_y - 1, foreground_col,
background_col);
    flood(seed_x - 1, seed_y + 1, foreground_col,
background_col);
  }
}
```

Z



Program: -

```
 /* Experiment No 6:-Implement Scan_line algorithm for polygon
filling */
/* Example:- Enter vertices:-3
x[0]=100,y[0]=50,x[1]=150,y[1]=100,x[2]=50,y[2]=100*/
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
\*Defining the structure to store edges-- -- -- -- -- -- -- --
-- -- -- -- -- -- -- -- -- -- -- -- -- -* /
    struct edge
{
  int x1;
  int y1;
  int x2;
  int y2;
  int flag;
};

void main()
{
  int gd = DETECT, gm, n, I, j, k;
  struct edge ed[10], temped;
  float dx, dy, m[10], x_int[10], inter_x[10];
  int x[10], y[10], ymax = 0, ymin = 480, yy, temp;
  initgraph(&gd, &gm, "c:\\turboc3\\bgi");
  /* Read the number of vertices of the polygon
  ------------------------------------------- */
  printf("Enter the number vertices of the graph: ");
  scanf("%d", &n);
  /* Read the vertices of the polygon and also find Ymax and
Ymin
  ------------------------------------------------------------
--- */
  printf("Enter the vertices: \n");
  for (I = 0; I < n; i++)
  {
    printf("x[%d] : ", i);
    scanf("%d", &x[i]);
    printf("y[%d] : ", i);
    scanf("%d", &y[i]);
    if (y[i] > ymax)
      ymax = y[i];
    if (y[i] < ymin)
      ymin = y[i];
```

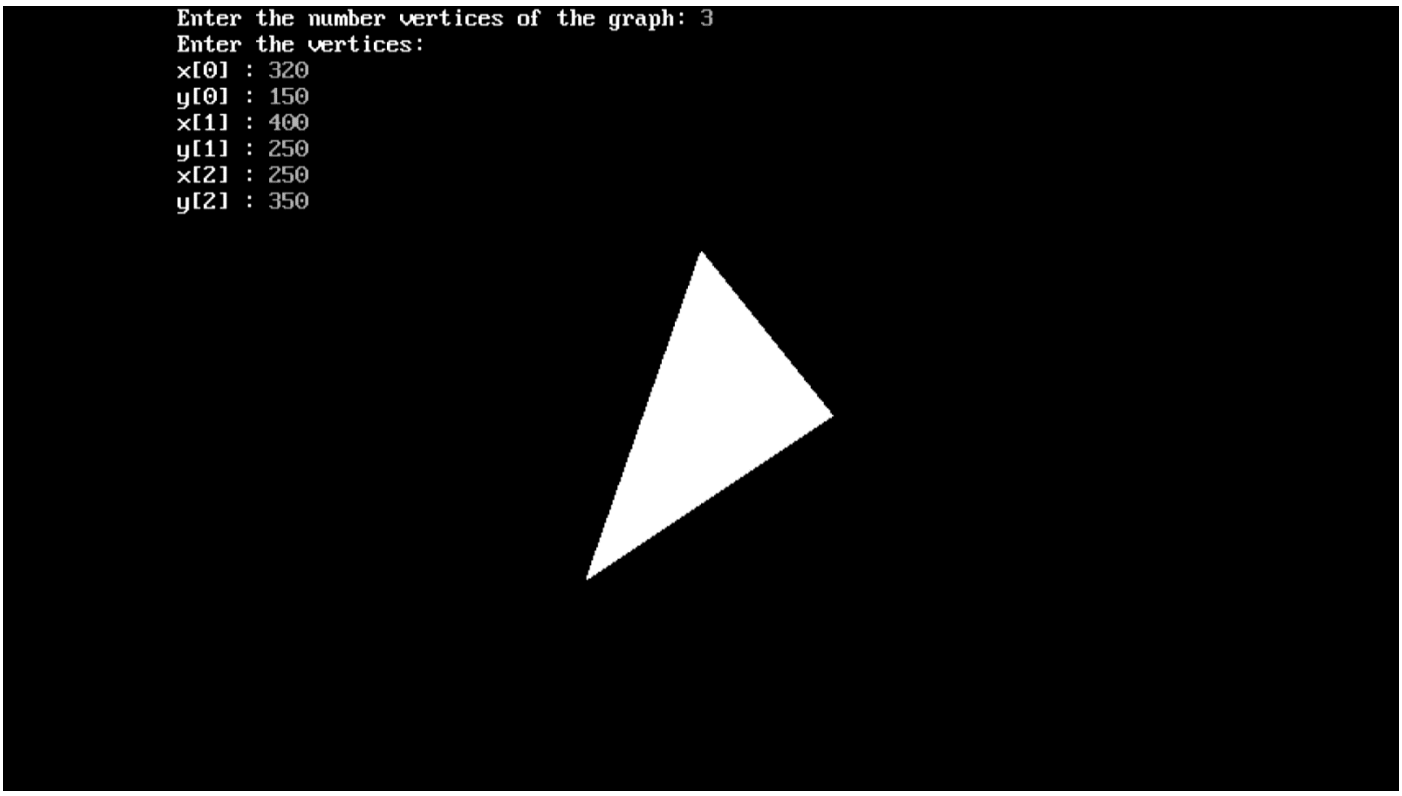
Z

```
  ed[i].x1 = x[i];
  ed[i].y1 = y[i];
}

/* Store the edge information
----------------------------*/
for (I = 0; I < n - 1; i++)
{
  ed[i].x2 = ed[I + 1].x1;
  ed[i].y2 = ed[I + 1].y1;
  ed[i].flag = 0;
}
ed[i].x2 = ed[0].x1;
ed[i].y2 = ed[0].y1;
ed[i].flag = 0;

/* Check for y1>y2, if not interchange y1 and y2
   with corresponding x1 and x2
   -------------------------------*/
for (I = 0; I < n; i++)
{
  if (ed[i].y1 < ed[i].y2)
  {
    temp = ed[i].x1;
    ed[i].x1 = ed[i].x2;
    ed[i].x2 = temp;
    temp = ed[i].y1;
    ed[i].y1 = ed[i].y2;
    ed[i].y2 = temp;
  }
}
/* Draw the polygon
---------------------- */
for (I = 0; I < n; i++)
{
  line(ed[i].x1, ed[i].y1, ed[i].x2, ed[i].y2);
}

/* sorting of edges in the order of y1,y2,x1
------------------------------------------- */
for (I = 0; I < n - 1; i++)
{
  for (j = 0; j < n - 1; j++)
  {
    if (ed[j].y1 < ed[j + 1].y1)
    {
```

```
      temped = ed[j];
      ed[j] = ed[j + 1];
      ed[j + 1] = temped;
    }
    if (ed[j].y1 == ed[j + 1].y1)
    {
      if (ed[j].y2 < ed[j + 1].y2)
      {
        temped = ed[j];
        ed[j] = ed[j + 1];
        ed[j + 1] = temped;
      }
      if (ed[j].y2 == ed[j + 1].y2)
      {
        if (ed[j].x1 < ed[j + 1].x1)
        {
          temped = ed[j];
          ed[j] = ed[j + 1];
          ed[j + 1] = temped;
        }
      }
    }
  }
}
/* calculating 1/slope of each edge  and storing top x
coordinate of the edge     -------------------------- */
for (I = 0; I < n; i++)
{
  dx = ed[i].x2 - ed[i].x1;
  dy = ed[i].y2 - ed[i].y1;
  if (dy == 0)
  {
    m[i] = 0;
  }
  else
  {
    m[i] = dx / dy;
  }
  inter_x[i] = ed[i].x1;
}

yy = ymax;
while (yy > ymin)
{
  /*  Marking active egdes
  ------------------------- */
```

```
    for (I = 0; I < n; i++)
    {
      if (yy > ed[i].y2 && yy <= ed[i].y1 && ed[i].y1 !=
ed[i].y2)
      {
        ed[i].flag = 1;
      }
      else
      {
        ed[i].flag = 0;
      }
    }

    /* Finding the x intersections
    ------------------------------  */
    j = 0;
    for (I = 0; I < n; i++)
    {
      if (ed[i].flag == 1)
      {
        if (yy == ed[i].y1)
        {
          x_int[j] = ed[i].x1;
          j++;
          if (ed[I - 1].y1 == yy && ed[I - 1].y1 < yy)
          {
            x_int[j] = ed[i].x1;
            j++;
          }
          if (ed[I + 1].y1 == yy && ed[I + 1].y1 < yy)
          {
            x_int[j] = ed[i].x1;
            j++;
          }
        }
        else
        {
          x_int[j] = inter_x[i] + (-m[i]);
          inter_x[i] = x_int[j];
          j++;
        }
      }
    }
    /* Sorting the x intersections
    ---------------------------------*/
    for (I = 0; I < j; i++)
```

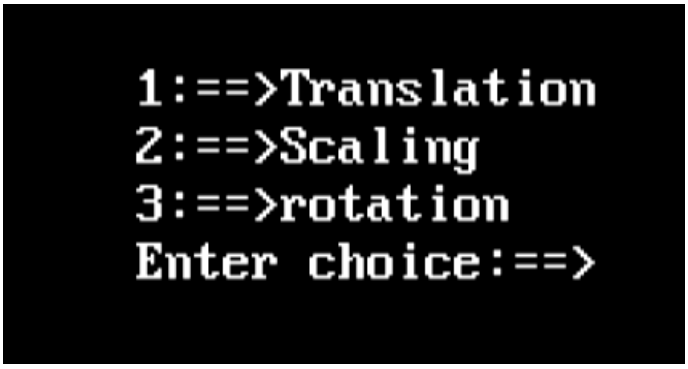
```
    {
      for (k = 0; k < j - 1; k++)
      {
        if (x_int[k] > x_int[k + 1])
        {
          temp = x_int[k];
          x_int[k] = x_int[k + 1];
          x_int[k + 1] = temp;
        }
      }
    }
    /* Extracting pairs of x values to draw lines
    ---------------------------------------------- */
    for (I = 0; I < j; I += 2)
    {
      line(x_int[i], yy, x_int[I + 1], yy);
    }
    yy--;
    delay(50);
  }
  getch();
}
```



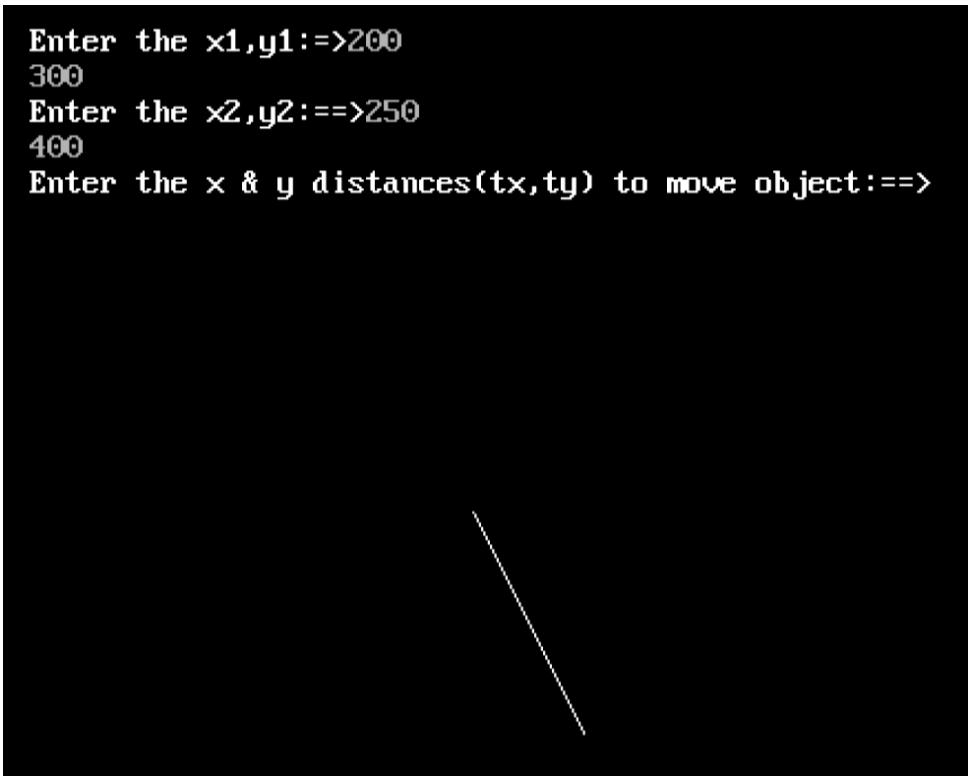
```
/*Experiment No7&8 :==Write  A Program for 2D Transformation
(Translation,Scaling,Rotation)*/
#include <stdio.h>
#include <graphics.h>
#include <math.h>
#include <conio.h>
#include <dos.h>
void main()
{
  int I, gd, gm, choice;
  char ch;
  void translation();
  void scaling();
  void rotation();
  clrscr();
  /* initialise graphics */
  detectgraph(&gd, &gm);
  initgraph(&gd, &gm, "c:\\turboc3\\bgi");
  do
  {
    printf("\n 1:Translation \n 2:Scaling \n 3:rotation");
    printf("\n Enter choice:");
    scanf("%d", &choice);
    switch (choice)
    {
    case 1:
      translation();
      break;
    case 2:
      scaling();
      break;
    case 3:
      rotation();
      break;
    }
    printf("\n Do youwant to Continue(y/n):");
    scanf("%s", &ch);
  } while (ch != 'n');
  getch();
  closegraph();
}
void translation()
{
  int x1, y1, x2, y2;
  int x11, y11, x22, y22;
```

```
  int tx, ty;
  /* Read the x1,y1,x2,y2 line endpoints */
  cleardevice();
  printf("Enter the x1,y1:=>");
  scanf("%d%d", &x1, &y1);
  printf("Enter the x2,y2:");
  scanf("%d%d", &x2, &y2);
  line(x1, y1, x2, y2);
  printf("Enter the x & y distances(tx,ty) to move object:");
  scanf("%d%d", &tx, &ty);
  x11 = x1 + tx;
  y11 = y1 + ty;
  x22 = x2 + tx;
  y22 = y2 + ty;
  line(x11, y11, x22, y22);
}
void scaling()
{
  int x1, y1, x2, y2;
  int x11, y11, x22, y22;
  int sx, sy;
  /* Read the x1,y1,x2,y2 line endpoints */
  cleardevice();
  printf("Enter the x1,y1:=>");
  scanf("%d%d", &x1, &y1);
  printf("Enter the x2,y2:");
  scanf("%d%d", &x2, &y2);
  line(x1, y1, x2, y2);
  printf("Enter the x & y distances to scale(sx,sy) object:");
  scanf("%d%d", &sx, &sy);
  x11 = x1 * sx;
  y11 = y1 * sy;
  x22 = x2 * sx;
  y22 = y2 * sy;
  line(x11, y11, x22, y22);
}
void rotation()
{
  int x1, y1, x2, y2;
  int x11, y11, x22, y22, a;
  float theta;
  /* Read the x1,y1,x2,y2 line endpoints- */
  cleardevice();
  printf("Enter the x1,y1:=>");
  scanf("%d%d", &x1, &y1);
  printf("Enter the x2,y2:==>");
```

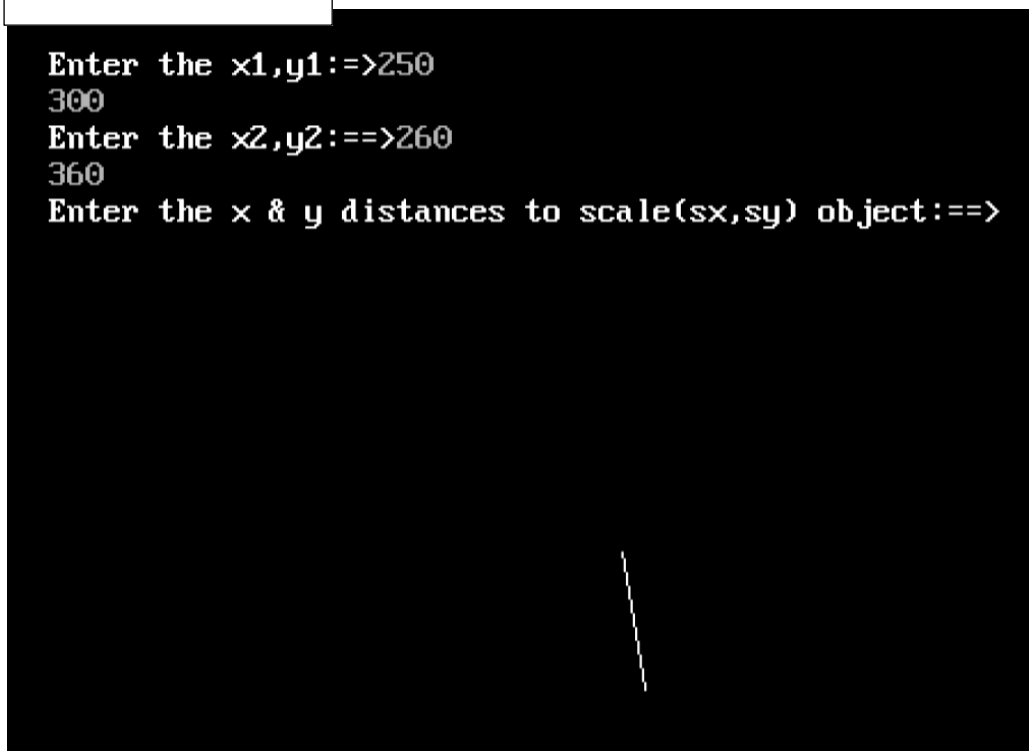
```
  scanf("%d%d", &x2, &y2);
  line(x1, y1, x2, y2);
  printf("Enter Rotation angle:==>");
  scanf("%d", &a);
  theta = ((a * 3.14) / 180);
  x11 = x1 * cos(theta) - y1 * sin(theta);
  y11 = x1 * sin(theta) + y1 * cos(theta);
  x22 = x2 * cos(theta) - y2 * sin(theta);
  y22 = x2 * sin(theta) + y2 * cos(theta);
  line(x11, y11, x22, y22);
}
```



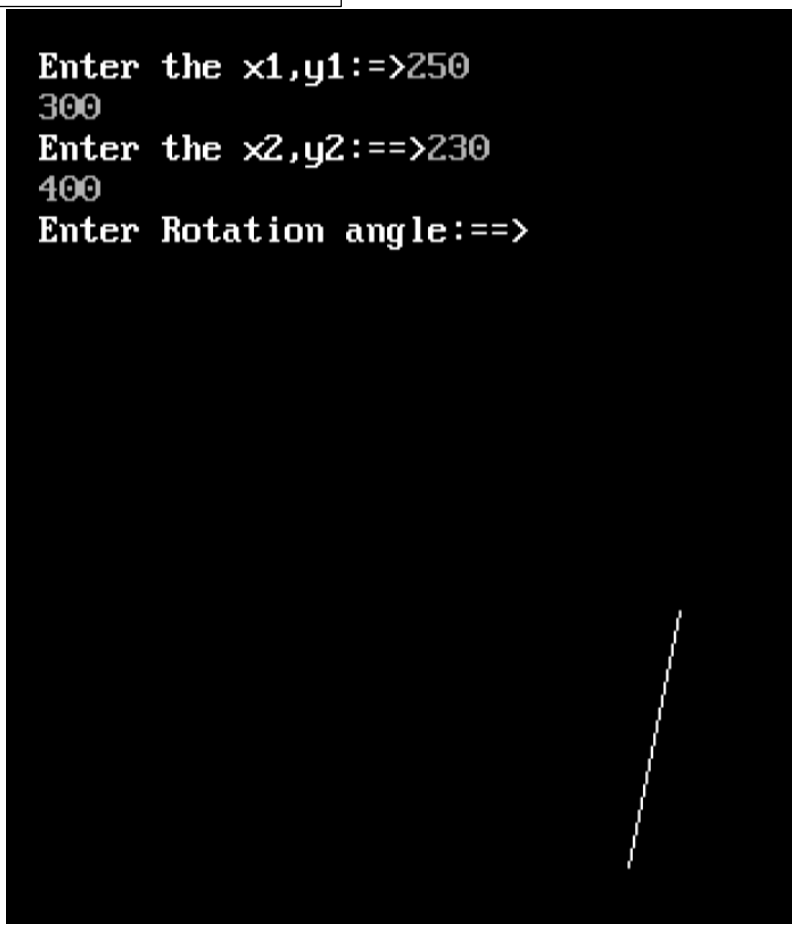
TRANSLATION



SCALLING



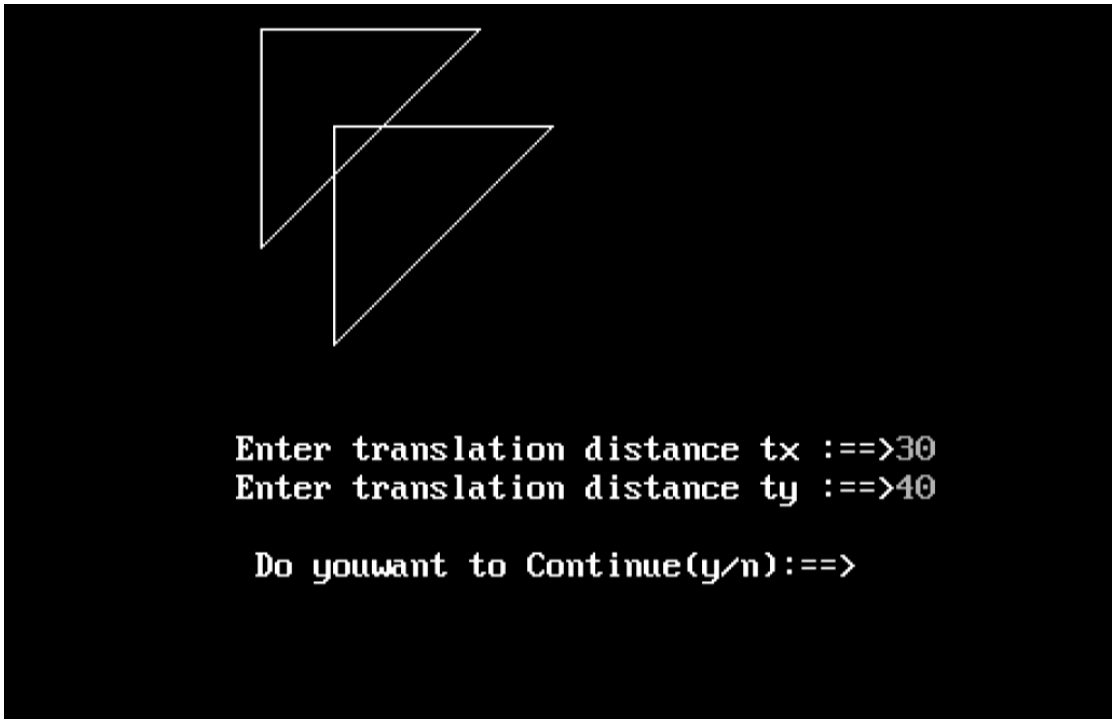
Rotation

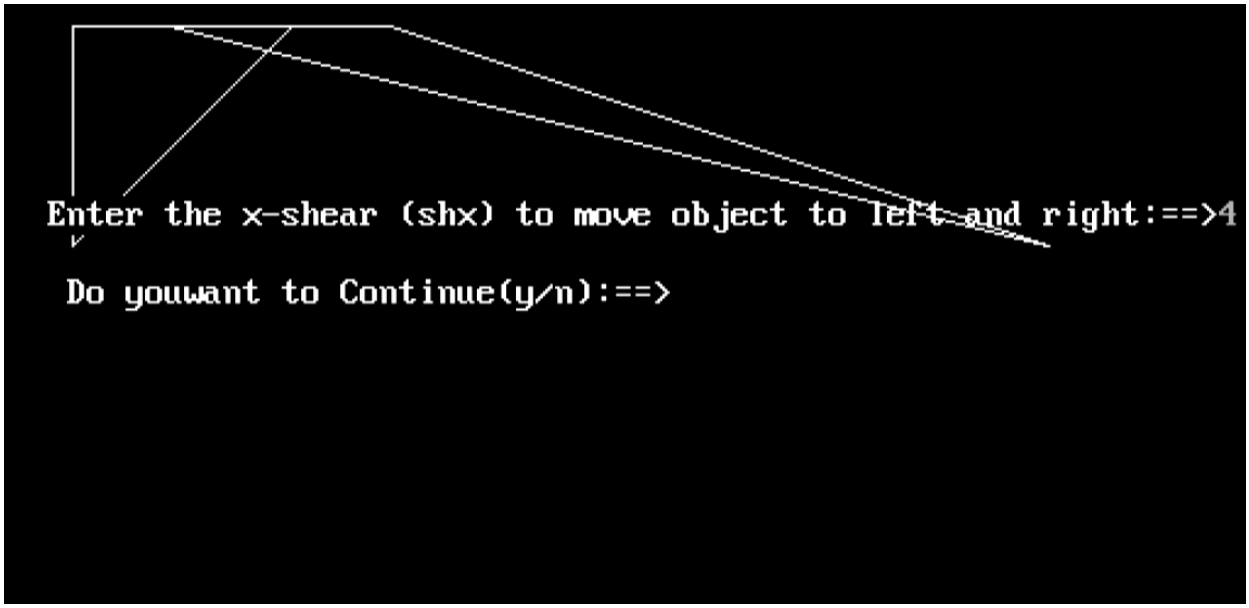


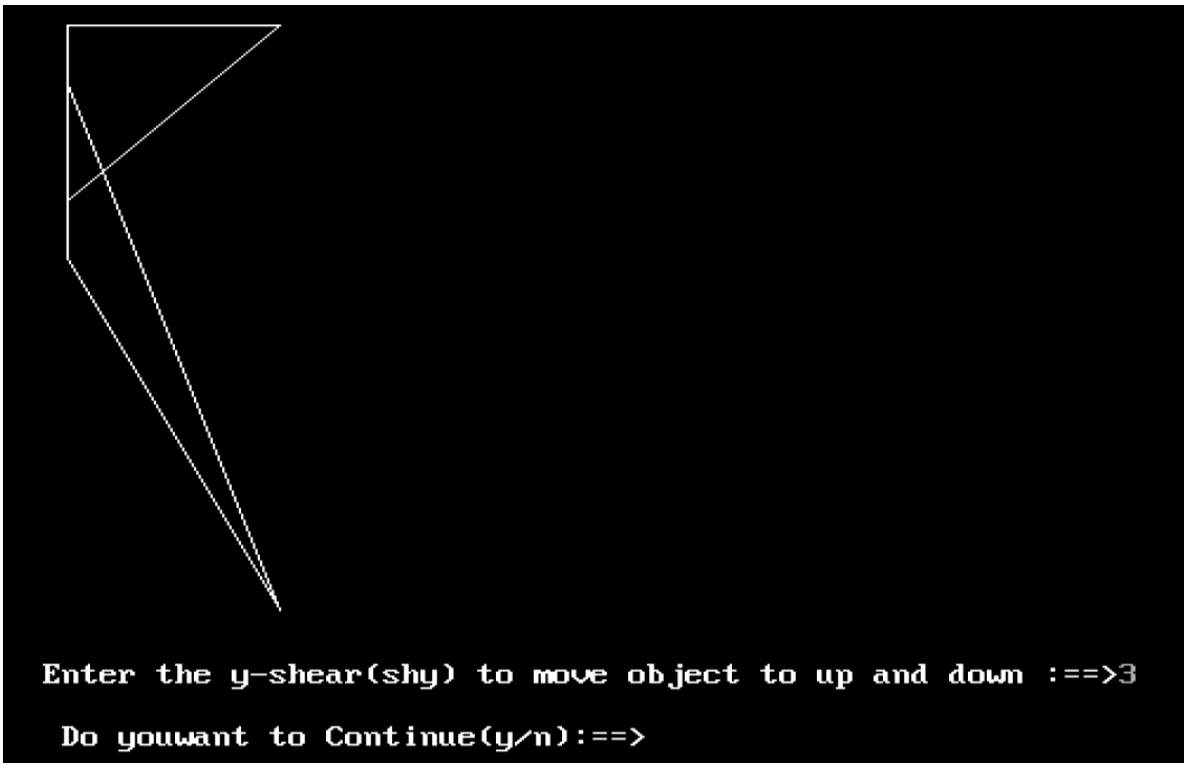
```
#include <stdio.h>
#include <graphics.h>
#include <math.h>
#include <conio.h>
#include <dos.h>
void main()
{
  int i, gd = DETECT, gm, choice;
  char ch;
  void translation();
  void x_shear();
  void y_shear();
  clrscr();
  /* initialise graphics
  ----------------------- */
  initgraph(&gd, &gm, "c:\\turboc3\\bgi");
  do
  {
    printf("\n 1:==>Translation \n 2:==>X-Shear \n 3:==>Y-
Shear \n ");
    printf("Enter choice:==>");
    scanf("%d", &choice);
    switch (choice)
    {
    case 1:
      translation();
      break;
    case 2:
      x_shear();
      break;
    case 3:
      y_shear();
      break;
    }
    printf("\n Do youwant to Continue(y/n):==>");
    scanf("%s", &ch);
  } while (ch != 'n');
  getch();
  closegraph();
}

void x_shear()
{
```

```
  float x1 = 10, y1 = 10, x2 = 100, y2 = 10, x3 = 10, y3 =
100;
  float x11, y11, x22, y22, x33, y33, tx, ty;
  float shx;
  cleardevice();
  line(x1, y1, x2, y2);
  line(x2, y2, x3, y3);
  line(x3, y3, x1, y1);
  printf("Enter the x-shear (shx) to move object to left and
right:==>");
  scanf("%f", &shx);
  x11 = x1 + shx * y1;
  y11 = y1;
  x22 = x2 + shx * y2;
  y22 = y2;
  x33 = x3 + shx * y3;
  y33 = y3;
  line(x11, y11, x22, y22);
  line(x22, y22, x33, y33);
  line(x33, y33, x11, y11);
}

void y_shear()
{
  float x1 = 10, y1 = 10, x2 = 100, y2 = 10, x3 = 10, y3 =
100;
  float x11, y11, x22, y22, x33, y33, tx, ty;
  float shy;
  cleardevice();
  line(x1, y1, x2, y2);
  line(x2, y2, x3, y3);
  line(x3, y3, x1, y1);
  printf("Enter the y-shear(shy) to move object to up and down
:==>");
  scanf("%f", &shy);
  x11 = x1;
  y11 = y1 + shy * x1;
  x22 = x2;
  y22 = y2 + shy * x2;
  x33 = x3;
  y33 = y3 + shy * x3;
  line(x11, y11, x22, y22);
  line(x22, y22, x33, y33);
  line(x33, y33, x11, y11);
}
```

```
void translation()
{
  int x1 = 10, y1 = 10, x2 = 100, y2 = 10, x3 = 10, y3 = 100;
  int x11, y11, x22, y22, x33, y33, tx, ty;
  cleardevice();
  line(x1, y1, x2, y2);
  line(x2, y2, x3, y3);
  line(x3, y3, x1, y1);
  printf("Enter translation distance tx :==>");
  scanf("%d", &tx);
  printf("Enter translation distance ty :==>");
  scanf("%d", &ty);
  x11 = x1 + tx;
  y11 = y1 + ty;
  x22 = x2 + tx;
  y22 = y2 + ty;
  x33 = x3 + tx;
  y33 = y3 + ty;
  line(x11, y11, x22, y22);
  line(x22, y22, x33, y33);
  line(x33, y33, x11, y11);
}
```




Enter the x-shear (shx) to move object to left and right:==>4
Do youwant to Continue(y/n):==>


Enter the y-shear(shy) to move object to up and down :==>3
Do youwant to Continue(y/n):==>