```
 /***   Program to Draw a Circle using Bresenham's Algorithm   ***/

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

void circleBres(int, int, int);
void drawCircle(int, int, int, int);

void main()
{
     int xc, yc, r;

     int gd = DETECT, gm;
     initgraph(&gd, &gm, "");

     printf("Enter center coordinates of circle: ");
     scanf("%d %d", &xc, &yc);
     printf("Enter radius of circle: ");
     scanf("%d", &r);

     circleBres(xc, yc, r);

     getch();
}

void circleBres(int xc, int yc, int r)
{
     int x = 0, y = r;
     int d = 3 - 2 * r;

     while (x < y)
     {
          drawCircle(xc, yc, x, y);
          x++;
```

```
            if (d < 0)
                  d = d + 4 * x + 6;
            else
            {
                  y--;
                  d = d + 4 * (x - y) + 10;
            }

            drawCircle(xc, yc, x, y);
            delay(50);
      }
}

void drawCircle(int xc, int yc, int x, int y)
{
      putpixel(xc+x, yc+y, RED);
      putpixel(xc-x, yc+y, RED);
      putpixel(xc+x, yc-y, RED);
      putpixel(xc-x, yc-y, RED);
      putpixel(xc+y, yc+x, RED);
      putpixel(xc-y, yc+x, RED);
      putpixel(xc+y, yc-x, RED);
      putpixel(xc-y, yc-x, RED);
}
```

```
 /***   Program to Draw a Circle using Mid - Point Algorithm   ***/

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

void circleMidpoint(int, int, int);
void drawCircle(int, int, int, int);

void main()
{
     int xc, yc, r;

     int gd = DETECT, gm;
     initgraph(&gd, &gm, "");

     printf("Enter center coordinates of circle: ");
     scanf("%d %d", &xc, &yc);
     printf("Enter radius of circle: ");
     scanf("%d", &r);

     circleMidpoint(xc, yc, r);

     getch();
}

void circleMidpoint(int xc, int yc, int r)
{
     int x = 0, y = r;
     int p = 1 - r;

     while (x < y)
     {
          drawCircle(xc, yc, x, y);
          x++;
```

```
            if (p < 0)
                 p = p + 2 * x + 1;
            else
            {
                 y--;
                 p = p + 2 * (x - y) + 1;
            }

            drawCircle(xc, yc, x, y);
            delay(50);
      }
}

void drawCircle(int xc, int yc, int x, int y)
{
      putpixel(xc+x, yc+y, RED);
      putpixel(xc-x, yc+y, RED);
      putpixel(xc+x, yc-y, RED);
      putpixel(xc-x, yc-y, RED);
      putpixel(xc+y, yc+x, RED);
      putpixel(xc-y, yc+x, RED);
      putpixel(xc+y, yc-x, RED);
      putpixel(xc-y, yc-x, RED);
}
```

```
 /***  Program to Draw an Ellipse using Mid - Point Algorithm  ***/

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

void ellipseMidpoint(float, float, float, float);
void drawEllipse(float, float, float, float);

void main()
{
     float xc, yc, rx, ry;

     int gd = DETECT, gm;
     initgraph(&gd, &gm, "");

     printf("\nEnter the center coordinates of ellipse: ");
     scanf("%f %f", &xc, &yc);
     printf("\nEnter x-radius coordinate: ");
     scanf("%f", &rx);
     printf("\nEnter y-radius coordiante: ");
     scanf("%f", &ry);

     ellipseMidpoint(xc, yc, rx, ry);

     getch();
}

void ellipseMidpoint(float xc, float yc, float rx, float ry)
{
     float rxSq = rx * rx;
     float rySq = ry * ry;
     float x = 0, y = ry, p;
     float px = 0, py = 2 * rxSq * y;
```

```
    drawEllipse(xc, yc, x, y);

  //Region 1
    p = rySq - (rxSq * ry) + (0.25 * rxSq);

    while (px < py)
    {
          x++;
          px = px + 2 * rySq;

          if (p < 0)
               p = p + rySq + px;
          else
          {
               y--;
               py = py - 2 * rxSq;
               p = p + rySq + px  - py;
          }

          drawEllipse(xc, yc, x, y);
          delay(30);
    }

  //Region 2
    p = rySq*(x+0.5)*(x+0.5) + rxSq*(y-1)*(y-1) - rxSq*rySq;

    while (y > 0)
    {
          y--;
          py = py - 2 * rxSq;

          if (p > 0)
               p = p + rxSq - py;
```

```
            else
            {
                  x++;
                  px = px + 2 * rySq;
                  p = p + rxSq - py + px;
            }

            drawEllipse(xc, yc, x, y);
            delay(30);
      }
}

void drawEllipse(float xc, float yc, float x, float y)
{
      putpixel(xc+x, yc+y, RED);
      putpixel(xc-x, yc+y, RED);
      putpixel(xc+x, yc-y, RED);
      putpixel(xc-x, yc-y, RED);
}
```

```
 /***    Program to Draw a Line using Bresenham's Algorithm    ***/

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

void lineBres(int, int, int, int);

void main()
{
     int x1, y1, xn, yn;

     int gd = DETECT, gm;
     initgraph(&gd, &gm, "");

     printf("Enter starting coordinates of line: ");
     scanf("%d %d", &x1, &y1);
     printf("Enter ending coordinates of line: ");
     scanf("%d %d", &xn, &yn);

     lineBres(x1, y1, xn, yn);

     getch();
}

void lineBres(int x1, int y1, int xn, int yn)
{
     int dx = xn - x1, dy = yn - y1;
     int di = 2 * dy - dx;
     int ds = 2 * dy, dt = 2 * (dy - dx);

     putpixel(x1, y1, RED);
```

```
      while (x1 < xn)
      {
            x1++;
            if (di < 0)
                  di = di + ds;
            else
            {
                  y1++;
                  di = di + dt;
            }

            putpixel(x1, y1, RED);
            delay(20);
      }
}
```

```
    /****    Program to Draw a Line using DDA Algorithm    ****/

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

void lineDDA(int, int, int, int);

void main()
{
     int x1, y1, xn, yn;

     int gd = DETECT, gm;
     initgraph(&gd, &gm, "");

     printf("Enter the starting coordinates of line: ");
     scanf("%d %d", &x1, &y1);
     printf("Enter the ending coordinates of line: ");
     scanf("%d %d", &xn, &yn);

     lineDDA(x1, y1, xn, yn);

     getch();
}

void lineDDA(int x1, int y1, int xn, int yn)
{
     int dx, dy, m, i;
     m = (yn-y1)/(xn-x1);

     for (i=x1; i<=xn; i++)
     {
          if (m <= 1)
          {
               dx = 1;
```

```
                dy = m * dx;
            }
            else
            {
                dy = 1;
                dx = dy / m;
            }

            x1 = x1 + dx;
            y1 = y1 + dy;

            putpixel(x1, y1, RED);
            delay(20);
        }
}
```