Name :→ Kanchan Sharma
University Roll No. :→ 1121070.
Course :→ BCA 6-C
Subject :→ Computer Graphics & Animation
(PBC-602).

P1: Bresenham's line Drawing Algorithm.

Algorithm: Given starting coordinates = (X_0, Y_0)
ending coordinates = (X_n, Y_n)

Step 1: Calculate ΔX and ΔY.

$$\Delta X = X_n - X_0$$
$$\Delta Y = Y_n - Y_0.$$

Step 2: Calculate decision parameter, P_k.

$$P_K = 2\Delta Y - \Delta X$$

Step 3: Suppose current point is (X_k, Y_k) and the next points is (X_{k+1}, Y_{k+1}).
find next point depending on P_k.

Case I: If $P_K < 0$

$$P_{K+1} = P_K + 2\Delta Y$$
$$X_{K+1} = X_K + 1$$
$$Y_{K+1} = Y_K$$

Case II : If $p_k >= 0$

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$
$$x_{k+1} = x_k + 1$$
$$y_{k+1} = y_k + 1.$$

Step 4: Keep repeating step-3 until the end point is reached or number of iterations equal to $(\Delta x - 1)$ times.

Program:

```
#include <stdio.h>
#include <graphics.h>
void drawline (int x0, int y0, int x1, int y1)
{
    int dx, dy, p, x, y;
    dx = x1-x0;
    dy = y1-y0;
    x= x0;
    y= y0;
    p= 2*dy-dx;
    while (x<x1)
    {
        if (p>=0)
        {
            putpixel (x, y, 7);
            y= y+1;
            p= p+2*dy;
        }
```

```
else
 {
    putpixel (x, y, 7);
    p = p + 2 * dy;
 }
 x = x + 1;
 }
}
int main()
{
   int gdriver = DETECT, gmode, error, x0, y0, x1, y1;
   initgraph (&gdriver, &gmode, " ");
printf (" enter first point coordinates : ");
scanf (" %d %d ", &x0, &y0);
printf (" enter second point coordinates : ");
scanf (" %d %d ", &x1, &y1);
 drawline (x0, y0, x1, y1);
return 0;
}
```

P2: Mid Point Circle Drawing Algorithm

Algorithm: Given,
centre of circle in point $=(X_0, Y_0)$
Radius of circle $= R$

Step 1: Assign the starting point coordinates (X_0, Y_0) as-

$X_0 = 0$
$Y_0 = R$

Step 2: Calculate the value of initial decision parameter as —

$P_0 = 1 - R$

Step 3: Suppose the current point is (X_K, Y_K) and the next point is (X_{K+1}, Y_{K+1})
Find the next point of first octant depending on the P_K.

Case I: If $P_K < 0$

$X_{K+1} = X_K + 1$

$Y_{K+1} = Y_K$

$P_{K+1} = P_K + 2 \times X_{K+1} + 1$

Case II: If $P_K >= 0$

$X_{K+1} = X_K + 1$

$Y_{K+1} = Y_K - 1$

$P_{K+1} = P_K - 2 \times Y_{K+1} + 2 \times X_{K+1} + 1$

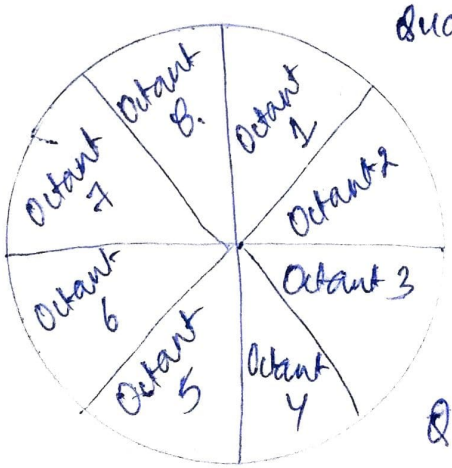
Step 4: If the given centre point (X_0, Y_0) is not $(0,0)$ then do the following steps:-

$$X_{plot} = X_c + X_0$$
$$Y_{plot} = Y_c + Y_0$$

Here (X_c, Y_c) denotes the current value of x and y coordinates.

Step 5: Keep repeating step 3 and step 4 until $X_{plot} >= Y_{plot}$.

Step 6: Step 5 generates all the points for one octant.



Quadrant 2
$(-X, Y)$

Quadrant 1
(X, Y)

Quadrant 3
$(-X, -Y)$

Quadrant 4
$(X, -Y)$

Program:

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
void main()
{
  int x, y, x_mid, y_mid, radius, dp;
  int g_mode, g_driver = DETECT;
  clrscr();
initgraph(&g_driver, &g_mode, "");
printf("Enter coordinates:");
scanf("%d%d", &x_mid, &y_mid);
printf("Enter radius:");
scanf("%d", &radius);
x = 0;
y = radius;
dp = 1 - radius;
do
{
  putpixel(x_mid + x, y_mid + y, YELLOW);
  putpixel(x_mid + y, y_mid + x, YELLOW);
  putpixel(x_mid - y, y_mid + x, YELLOW);
  putpixel(x_mid - x, y_mid + y, YELLOW);
  putpixel(x_mid - x, y_mid - y, YELLOW);
```

```
putpixel (x_mid-y y_mid-x ex, YELLOW);
putpixel (x_mid+y y_mid-x ecx, YELLOW);
putpixel (x_mid+x y_mid-y , YELLOW);
  if (dp<0)
  {
     dp+= (2*x)+1;
  }
  else
  {
     y= y-1;
    dp+=(2*x)-(2*y)+1;
  }
  x= x+1;
 }
   while (y>x);
 getch();
}
```