Name: Himanshu Prakash
Roll no: 11121062 (06)
Subject: Computer graphics

End term Practical

Program 1 Bresenham line algo.

```
#include<stdio.h>
#include<graphic.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 - 2*dx;
      }
      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 co-ordinates of first point");
 scanf ("%d %d", &x0, &y0);
Pritf ("Enter co-ordinates of second point:");
 Scant ("%d %d", &x1, &y1);
 drawline (x0, y0, x1, y1);

 return 0;
}
```

# Algo of Bresenham line Algorithm

Step1: Start Algorithm

Step2: Declare variable $x_1, x_2, y_1, y_2, d, i_1, i_2, dx, dy$

Step 3: Enter Value of $x_1, y_1, x_2, y_2$

where $x_1, y_1$ are Starting point

$x_2, y_2$ are ending point

Step4: calculate $dx = x_2 - x_1$

$dy = y_2 - y_1$

$i_1 = 2 * dy$

$i_2 = 2 * (dy - dx)$

$d = i_1 - dx$

Step5: Consider $(x, y)$ as starting point $x_{end}$ as Maximum possible value of $x$.

if $dx < 0$ then $x = x_2$, $y = y_2$

$x_{end} = x_1$

if $dx > 0$

then $x = x_1$

$y = y_1$

$x_{end} = x_2$

Step6: Generate point at $(x, y)$ Cordinates

Step 7 : Check if whole line is generated

if $x >= x_{end}$.

Stop

Step 8 : Calculate co-ordinates of the next pixel

if $d < 0$

then $d = d + i_1$

if $d \geq 0$

Then $d = d + i_2$

Increment $y = y + 1$

Step 9 : Increment $x = x + 1$

Step 10 : Draw a point of latest $(x, y)$ Cordinate

Step 11 : Go to step 7

Step 12 : End of Algorithm.

Program2 Midpoint Circle

```
#include <stdio.h>
#include<graphic>
void drawcircle (int x0, int y0, int radius)
{
  int x = radius;
  int y = 0;
  int err = 0;
  while (x >= y)
  {
  Putpixel (x0 + x, y0 + y, 7);
  Putpixel (x0 + y, y0 + x, 7);
  Putpixel (x0 - y, y0 + x, 7);
  Putpixel (x0 - x, y0 + y, 7);
  Putpixel (x0 - x, y0 - y, 7);
  Putpixel (x0 + y, y0 - x, 7);
  Putpixel (x0 + x, y0 y, 7);
  Putpixel (x0 - y, y0 - x, 7);
  if (err <= 0)
  {
    y += 1;
    err += 2*y + 1;
  }
  if (err > 0)
  {
    x -= 1;
    err -= 2*x + 1;
  }
  }
}
```

```
int main()
{
int gdriver = DETECT, gmode, x, y, r;
initgraph (&gdriver, &gmode, " ")
Printf("Enter radius of circle:");
Scanf ("%d", &r);
Printf ("Enter Co-ordinates of Center(x and y):");
Scanf ("%d %d", &x, &y);
drawcircle (x, y, r);
return 0;
}
```

Prg2 Mid Point Circle Drawing 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
$x_0 = 0 \quad y_0 = R$

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 $P_k < 0$
$x_{k+1} = x_{k+1}$
$y_{k+1} = y_k$
$P_{k+1} = P_k + 2x_{k+1} + 1$

Case II : if $P_k >= 0$
$x_{k+1} = x_{k+1}$
$y_{k+1} = y_{k-1}$
$P_{k+1} = P_k - 2y_{k+1} + 2x_{k+1} + 1$

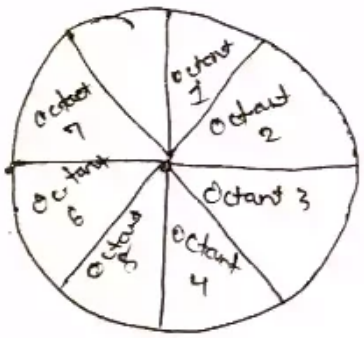
Step 4: If the given Centre point $(x_0, y_2)$ is not $(0, 0)$ the do the following Steps :-
$x_{plot} = x_c + x_0$
$x_{plot} = y_2 + y_0$
Here $(x_2, y_0)$ denotes the current value of x and y Coordinate

Step 5 :- Keep repeating Step 3 and Step 4 until

X plot >= Y plot

Step 6 : Step 5 generated all the points for one Octant.

Quadrant 2
(-X, Y)




Quadrant 1
(X, Y)

Quadrant 3
(-X, -Y)

Quadrant 4
(X, -Y)