Name- Sumit Rawat
Roll No.- 1121149
Subject - Computer Graphics
Course - BCA (6th sem)
Section- C SET - C

Q1) Write an Algorithm and Program to implement Breshenham Line Drawing Algorithm.

Sol: Bresenham's Line Algorithm

step 1: Start Algorithm

step 2: 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 coordinates of starting point
and x_2, y_2 are coordinates of Ending point

step 4: Calculate $dx = x_2 - x_1$
Calculate $dy = y_2 - y_1$
Calculate $i_1 = 2 * dy$
Calculate $i_2 = 2 * (dy - dx)$
Calculate $d = i_1 - dx$

step 5: consider (x, y) as starting point and 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 $x = x_1$
$y = y_1$
$x_{end} = x_2$

Sumit

Step 6: Generate point at (x, y) coordinates.

Step 7: check if whole line is generated
if $x >= x_{end}$
stop.

Step 8: Calculate co-ordinates of next pixel
if $d < 0$
then $d = d + i_1$
if $d \geq 0$
then $d = d + i_2$
increment $x = x + 1$

Step 9: increment $x = x + 1$

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

Step 11: Go to step 7

Step 12: End of Algorithm

Program to implement Bresenham's line Algorithm

```
#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 -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, "C:\\turbo C3\\bgi");
 Printf ("enter co-ordinates of first point");
Scanf ("%d %d", & x0, & y0);
Printf ("Enter co-ordinates of second point");
 Scanf ("%d %d", & x1, & y1);
drawline (x0, y0, x1, y1);
 return 0;
}
```

NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program:

```
Enter co-ordinates of first point: 100
100
Enter co-ordinates of second point: 200
200
```

Name - Sumit Rawat
Roll No. - 1122149
Subject - Computer Graphics
Course - BCA (6th Sem)
Section : C

Q2 ÷ Write An algorithm & program to implement mid-point circle Drawing algorithm.

Sol ÷ Mid-point Circle Algorithm

Step 1: Get Radius and coordinates from user

Step 2: Find out the decison parameter that decides the nearest point to select using:
$d = 5/4 - r$

Step 3: while y is greater than x do
- if d is smaller than 0, then
  $y = y$
  $x = x + 1$
  $d = 2x + 1$
- else
  $y = y - 1$
  $x = x + 1$
  $d = d + 2x - 2y + 1$

Step 4: Determine and plot the symmetry points for all eight octants.

Step 5: Repeat step 3 & step 4, till $y > x$

Sumit

C Program to ~~draw a circle~~ circle algorithm.

```
#include <conio.h>
#include <graphics.h>
void main()
{
int x, y, x mid, y mid, r, d;
int g_mode, g_driver = DETECT;
clrscr();
initgraph(&g_driver, &g_mode, "C:\\TURBOC3\\BGI");
~~Printf("MID Point Drawing algorithm \n\n");~~
Printf("\enter the coordinates");
Scanf("%d %od", &x_mid, &y_mid);
Printf("\n now enter radius =");
Scanf("%od", &r);
x = 0;
y = r;
d = 1-r;
do
{
Putpixel (x_mid + x, y_mid + y, 1);
Putpixel (x_mid + y, y_mid + x, 1);
Putpixel (x_mid - y, y_mid + x, 1);
Putpixel (x_mid - x, y_mid + y, 1);
Putpixel (x_mid - x, y_mid - y, 1);
Putpixel (x_mid - y, y_mid - x, 1);
Putpixel (x_mid + y, y_mid - x, 1);
Putpixel (x_mid + x, y_mid - y, 1);
if (d < 0)
{
d += (2 * x) + 1;
}
```

```
else
{
 y = y -1;
d  + (2* x)-(2 * y)+1;
 }
 x  =x +1 ;
 }
while (y >x);
 getch ();
 }
```

NeuTroN DOS-C++ 0.77, Cpu speed: max 100% cycles, Frameskip 0, Program:

```
Enter radius of circle: 100
Enter co-ordinates of center(x and y): 150
150
```