Name - Sachin Singh
Course - BCA
Sec - B
Roll no - 24(1121115)
Subj - computer graphic

Ans 3 > Bresenham's Circle Algorithm

Step 1: Start Algorithm
Step 2: Declare p, q, x, y, r, d variables p, q are coordinates of the center of the circle r is the radius of the circle
Step 3: enter the value of r
step 4: calculate d = 3-2r
step 5: initialize x=0 and nbay = r

step 6: check if
x >= y
stop

Step 7: Plot eight points by using concepts of eight-way symmetry. The center is at (p, q). current active pixel is (x, y)

putpixel(x+p, y+q)
putpixel(y+p, x+q)
putpixel(-y+p, x+q)
putpixel(-x+p, y+q)
putpixel(-x+p, -y+q)
putpixel(-y+p, -x+q)
putpixel(y+p, -x+q)
putpixel(x+p, -y-q)

Step 8:
Find location of next pixels to be scanned
if $d < 0$
then $d = d + 4x + 6$
increment $x = x + 1$
if $d \geq 0$
then $d = d + 4(x - y) + 10$
increment $x = x + 1$
decrement $y = y - 1$
Step 9: Go to step 6
Step 10: Stop also

Program

```
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
#include<math.h>

void Eightwaysymmetricplot(int xc, int yc, int x, int y)
{
putpixel(xc+x, yc+y, RED);
putpixel(xc+x, yc-y, YELLOW);
putpixel(xc-x, yc-y, GREEN);
putpixel(xc-x, yc+y, YELLOW);
putpixel(xc+y, yc+x, 12);
putpixel(xc-y, yc+x, 15);
putpixel(xc-y, yc-x, 6);
}
Void Bresenhamcircle(int xc, int yc, int r)
{
int x=0, y=r, d=3-(2*r);
Eightwaysymmetricplot(xc, yc, x, y);
while(x<=y)
{
if(d<=0)
{
d=d+(4*x)+6;
```

```
else
{
d = d + (y+=c) - (4 * y) + 10;
y = y - 1;
}
x = x + 1;
Eightway(xc, yc, x, y);
}
}
int main(void)
{
/* request auto detection */
int xc, yc, r, gdriver = DETECT, gmode, errorcode;
initgraph(&gdriver, &gmode, "C:\\TURBOC3\\BGI");
errorcode = graphresult();
if(errorcode != grOk)
{
printf("Graphics error : %s\n", grapherrormsg(errorcode));
printf("Press any key to halt:");
getch();
exit(1);
}
printf("Enter the values of xc and yc ");
scanf("%d%d", &xc, &yc);
printf("enter the value of radius:");
scanf("%d", &r);
Wireenhancircle(xc, yc, r);
getch();
closegraph();
return 0;
}
```

Pgm 1

```
#include<stdio.h>
#include<conio.h>
#include<dos.h>
void flood(int,int,int,int);
void main()
{
int gd,gm=DETECT;
clrscr();
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\turboC3\BGI");
rectangle(50,50,100,100);
flood(55,55,12,0);
getch();
}
void flood(int x,int y,int fill_col,int old_col)
{
if(getpixel(x,y)==old_col)
{
delay(10);
putpixel(x,y,fill_col);
flood(x+1,y,fill_col,old_col);
flood(x-1,y,fill_col,old_col);
flood(x,y+1,fill_col,old_col);
flood(x,y-1,fill_col,old_col);
flood(x+1,y-1,fill_col,old_col);
flood(x+1,y+1,fill_col,old_col);
flood(x-1,y-1,fill_col,old_col);
flood(x-1,y+1,fill_col,old_col);
}
}
```

Algorithm

Step 1 - Initialize the value of seed point (seedx, seedy), fcolor and dcol

Step 2 - Define the boundary values of the polygon

Step 3 - Check if the current seed point is of default color then repeat the steps 4 and 5 till the boundary pixels reached

If getpixel(x,y) = dcol then repeat 4 and 5

Step 4 - change the default color with the fill color at the seed point

setPixel(seedx, seedy, fcol)

Step 5 - Recursively follow the procedure with four neighbour points

FloodFill(seedx - 1, seedy, fcol, dcol)
FloodFill(seedx + 1, seedy, fcol, dcol)
FloodFill(seedx, seedy - 1, fcol, dcol)
FloodFill(seedx + 1, seedy + 1, fcol, dcol)
FloodFill(seedx + 1, seedy - 1, fcol, dcol)
FloodFill(seedx - 1, seedy - 1, fcol, dcol)

Step 6 - exit