Rajesh Sharma
Roll no. 1121107 (26)
BCA sec. B (6th sem.

Ans. 1) Algorithm to implement 8-connected flood fill

Step 1 - Start

Step 2 - Draw the rectangle using rectangle function

Step 3 - Implement 8 connected flood fill with the co-ordinates x and y.

putpixel (x, y, newcol);

floodfill (x+1, y, old, newcol);

floodfill (x-1, y, old, newcol);

floodfill (x+1, y, old, newcol);

floodfill (x-1, y, old, newcol);

floodfill (x, y+1, old, newcol);

floodfill (x, y-1, old, newcol);

floodfill (x+1, y+1, odd, newcol);

floodfill (x-1, y+1, odd, newcol);

floodfill (x+1, y-1, odd, newcol);

floodfill (x-1, y-1, old, newcol);

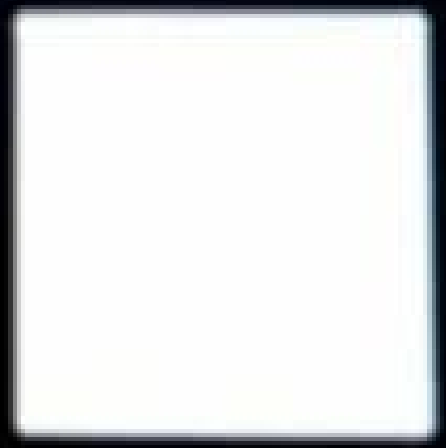
Step 4) Stop.

Program -

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

void floodfill ( int x, int y, int old, int newcol )
{
    int current;
    current = getpixel (x,y);
    if ( current == old )
    {
      delay (5);
      putpixel ( x,y, newcol );
      floodfill (x+1, y, old, newcol);
      floodfill (x-1, y, old, newcol);
      floodfill ( x, y+1, old, newcol );
      floodfill ( x, y-1, old, newcol);
      floodfill ( x+1, y+1, old, newcol);
      floodfill ( x-1, y+1, old, newcol);
      floodfill ( x+1, y-1, old, newcol);
      floodfill (x-1, y-1, old, newcol);
    }
}
```

```
int main()
{
  int gd = DETECT , gm ;
 initgraph ( &gd , &gm , "" ) ;
  rectangle ( 50 , 50 , 150 , 150 ) ;
  floodfill ( 70 , 70 , 0 , 15 ) ;
    getch ()
    closegraph () ;
}
```



Ans 3) Bresenham's Circle Drawing Algorithm.

Step 1 :- Start

step 2 - Declare p, q, x, y, r, d variable

p, q are coordinates of the centre 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 $x = 0$

$y = r$

Step 6 - check if the whole circle is scan converted.

if $x >= y$

stop.

Step 7 - Plot eight points by using concepts of eight way symmetry. The centor is at (p, q). Currunt 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.

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 (x+xc, y+yc, RED);
  putpixel (x+xc, -y+yc, YELLOW);
  putpixel (-x+xc, -y+yc, GREEN);
  putpixel (-x+xc, y+yc, YELLOW);
  putpixel (y+xc, x+yc, 12);
  putpixel (y+xc, -x+yc, 14);
```

```
putpixel (-y + xc, -x + yc, 15);
putpixel (-y + xc, x + yc, 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 + (4 * x) - (4 * y) + 10;
  y = y - 1;
  }
  x = x + 1;
  EightWaySymmetricPlot (xc, yc, x, y);
}
}
int main ()
{
int xc, yc, r, g, driver = DETECT, gmode, errorcode;
initgraph (&gdriver, &gmode, " ");
errorcode = graphresult ();
```

```
if( errorcode != grOk)
{
 printf("Graphic Error : %s\n", grapherrormsg(errorcode));
 printf(" Press any Key to halt);
  getch()
  exit(1);
}
printf(" Enter the value of xc & yc;
scanf("%d %d ,&xc, &yc);
printf(" Enter the value of radius :");
 scanf("%d", &r);
  BresenhamCircle( xy, yc, r);
  getch();
  closegraph();
  return 0;
}
```

```
Enter the values of xc and yc :100 100
Enter the value of radius  :50
```