Practical 1-A

a) Study and enlist the basic functions used for graphics in C / C++ / Python language. Give an example for each of them.

1) Arc function in C:

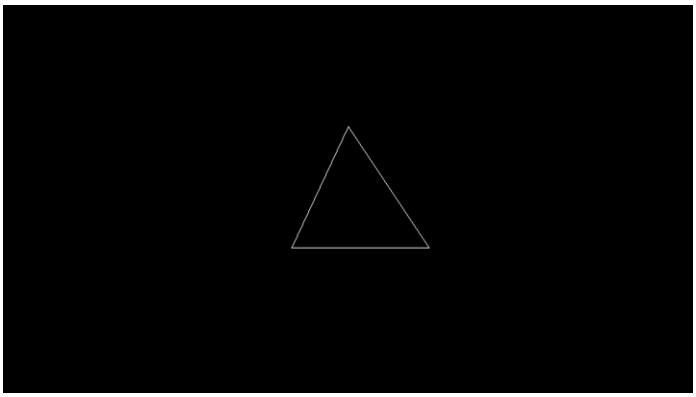
```
#include<graphics.h>
#include<conio.h>
main()
{
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "c:\\turboc3\\bgi");
    arc(100,100,0,135,50);
    getch();
    closegraph();
    return 0;
}
```

Output:-

2) Drawpoly function in C:

```
#include<graphics.h>
#include<conio.h>
main()
{
        int gd = DETECT, gm, points[]={320,150,420,300,250,300,320,150};
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        drawpoly(4,points);
        getch();
        closegraph();
        return 0;
}
```

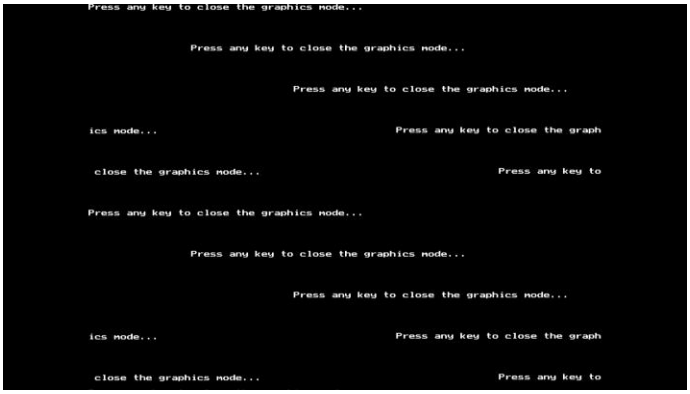
Output:-



3) Closegraph function in C:

```
#include<graphics.h>
#include<conio.h>
main()
{
        int gd = DETECT, gm;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        outtext("Press any key to close the graphics mode....);
        getch();
        closegraph();
        return 0;
}
```

Output:-

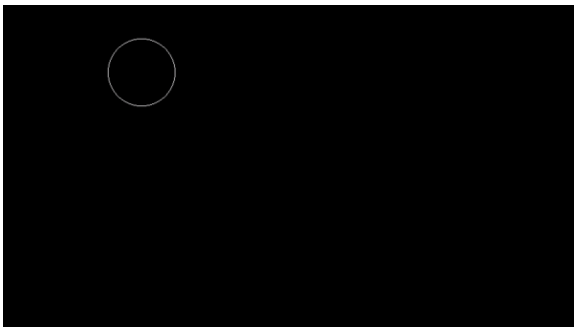


4) Circle function in C:

```
#include<graphics.h>
#include<conio.h>
main()
{
```

```
        int gd = DETECT, gm;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        circle(100,100,50);
        getch();
        closegraph();
        return 0;
}
```

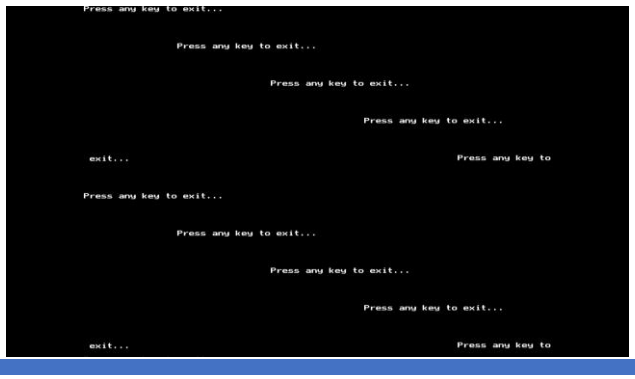
Output:-



5) Cleardevice function in C:

```
#include<graphics.h>
#include<conio.h>
main()
{
        int gd = DETECT, gm;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        outtext("Press any key to close the graphics mode...");
        getch();
        cleardevice();
        outtext(Press any key to exit....");
        getch();
        closegraph();
        return 0;
}
```

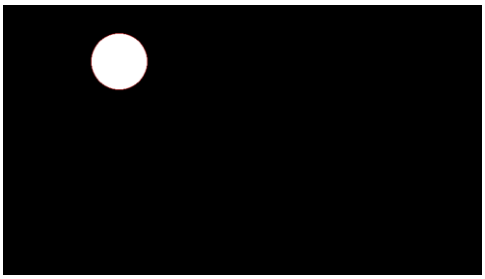
Output:-



6) Floodfill function:

```
#include<graphics.h>
#include<conio.h>
main()
{
        int gd = DETECT, gm;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        setcolor(RED);
        circle(100,100,50);
        floodfill(100,100,RED);
        getch();
        closegraph();
        return 0;
}
```

Output:-



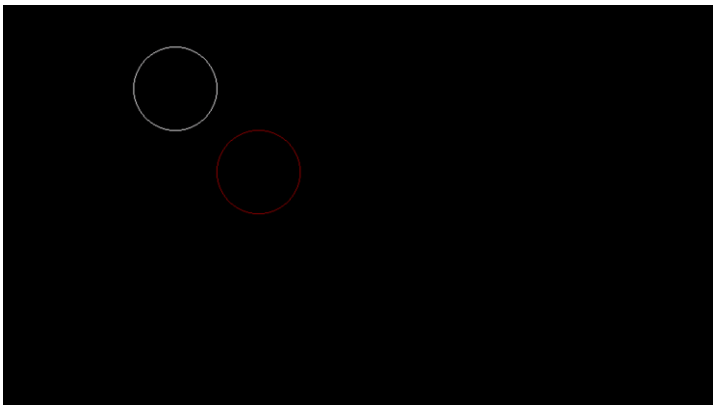
7) Getpixel function:

```
#include<graphics.h>
#include<conio.h>
main()
{
        int gd = DETECT, gm, color;
        char array[50];
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        color = getpixel(0,0);
        printf(array,"color of pixel at (0,0) = %d",color);
        outtext(array);
        getch();
        closegraph();
        return 0;
}
```

8) Setcolor function in C:

```
#include<graphics.h>
#include<conio.h>
main()
{
        int gd = DETECT, gm;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        circle(100,100,50);   /* drawn in white color */
        setcolor(RED);
        circle(100,100,50);   /* drawn in red color*/
        getch();
        closegraph();
        return 0;
}
```

Output:-



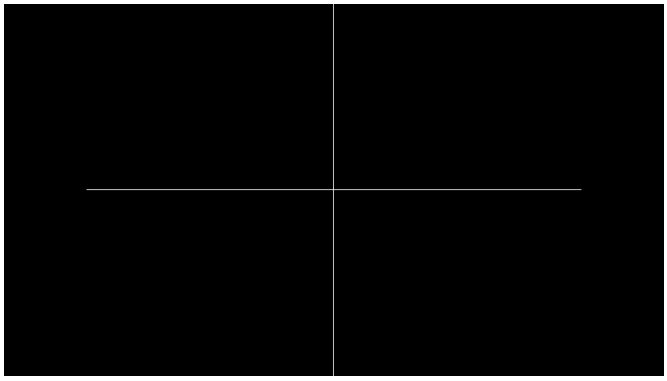
Practical 1-B

b) Draw a co-ordinate axis at the centre of the screen.

```
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
main()
{
        int gd = DETECT, gm,m,n;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        setcolor(getmaxcolor());
        m=getmaxx();
        n=getmaxy();
        line(m/2,0,m/2,n)
        line(0,n/2,m,n/2);
```

```
        getch();
        closegraph();
        return 0;
}
```

Output:-



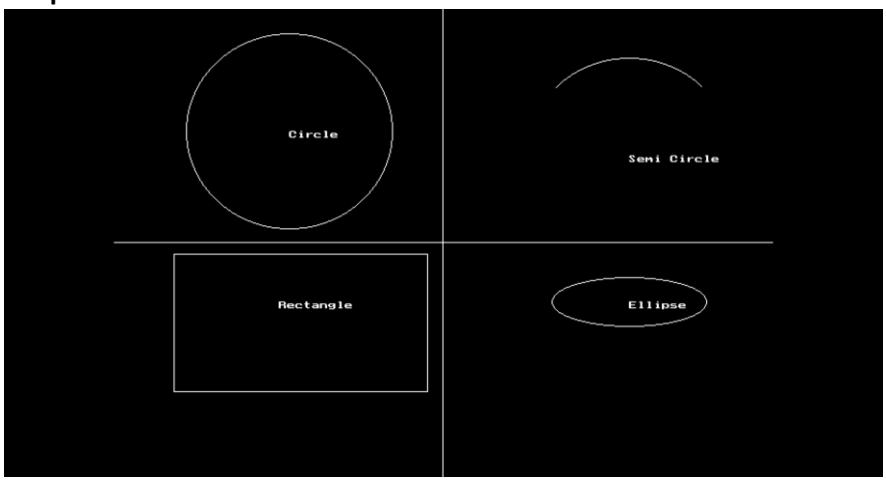
Practical 2

a) **Divide your screen into four region, draw circle, rectangle, ellipse and half ellipse in each region with appropriate message**.

```
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
main()
{
        int gd = DETECT, gm,m,n;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        setcolor(getmaxcolor());
        m=getmaxx();
        n=getmaxy();
        line(m/2,0,m/2,n);
        line(0,n/2,m,n/2);
        circle(170, 125, 100);
                outtextxy(170,125,"Circle");
        rectangle(58,251,304,392);
                outtextxy(160, 300, "Rectangle");
        arc(500, 150, 45, 135, 100);
                outtextxy(500,150,"Semi Circle");
        ellipse(500, 300, 0, 360, 75, 25);
                outtextxy(500,300, "Ellipse");
        getch();
        closegraph();
        return 0;
```

```
}
```

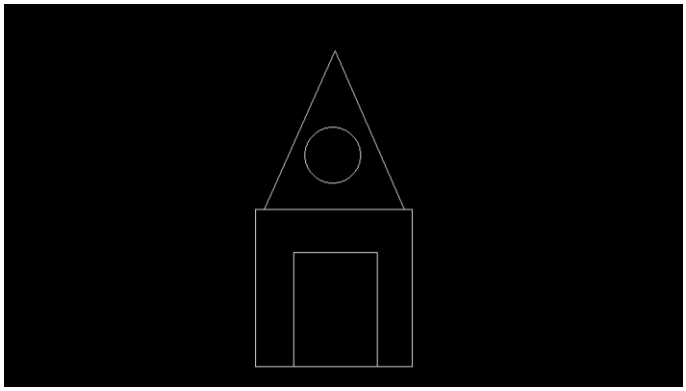
Output:-



b) Draw a simple hit on the screen.

```
/* Simple Hut on the Screen */
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
main()
{
        int gd = DETECT, gm,m,n;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        setcolor(getmaxcolor());
        m=getmaxx();
        n=getmaxy();
        rectangle(209,257,406,454);
        rectangle(257,311,362,454);
        line(309,58,220,257);
        line(309,58,396,257);
        circle(306,189,35);
        getch();
        closegraph();
        return 0;
}
```

Output:-



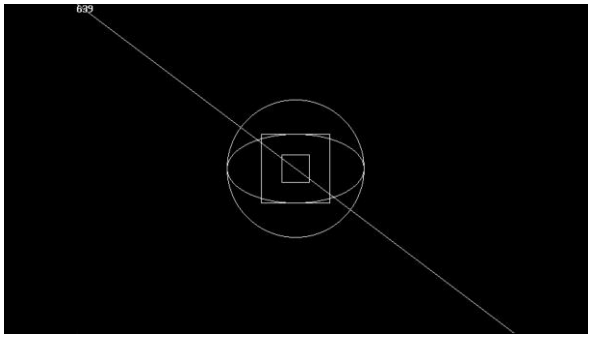
Practical 3

a) Draw the following basic shapes in the centre of the screen: (i) Circle, (ii) Rectangle, (iii) Square, (iv) Ellipse, (v) Line.

```
/* Basic Shapes in the centre */
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
main()
{
        int gd = DETECT, gm,errorcode;
        int midx, midy, left, top, right, bottom;
        int radius = 100;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        printf("%d", getmaxx());
        midx = getmaxx() / 2;
        midy = getmaxy() / 2;
        setcolor(getmaxcolor());
        circle(midx, midy, radius);
        getch();
        left = getmaxx() / 2 - 50;
        top = getmaxy() / 2 - 50;
        right = getmaxx() / 2 + 50;
        bottom = getmaxy() / 2 + 50;
        /* draw a rectangle*/
        rectangle(left, top, right, bottom);
        getch();
        rectangle(midx - 20, midy - 20, midx + 20, midy + 20);
        ellipse(midx, midy, 0, 360, 100, 50);
        getch();
        line(0, 0, midx * 2, midy * 2);
```

```
        getch();
        closegraph();
        return 0;
}
```

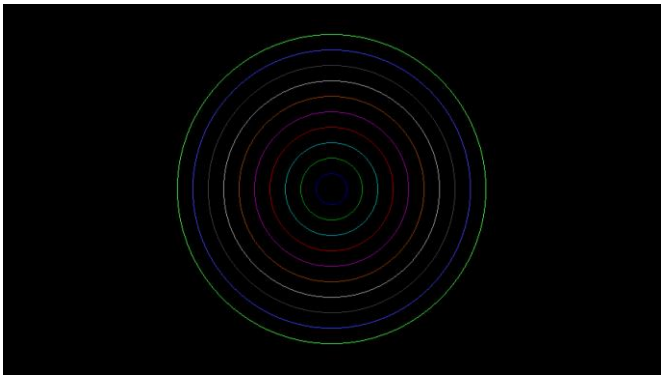
Output:-



b) Draw the following Concentric Circles in the Centre of the screen.

```
/* Concentric Circles */
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<conio.h>
main()
{
        int gd = DETECT, gm, color = 1, i;
        int midx, midy;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        midx = getmaxx() / 2;
        midy = getmaxy() / 2;
        for(i=20; i<=200; i+=20)
        {
                setcolor(color++);
                circle(midx, midy, i);
        }
        getch();
        closegraph();
        return 0;
}
```

Output:-



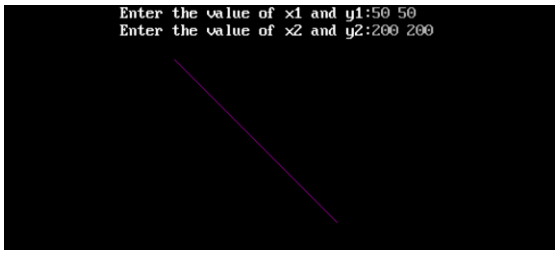
Practical 4

a) Develop the program for DDA Line drawing algorithm.

```
/* DDA Algorithm */
#include<graphics.h>
#include<iostream.h>
#include<math.h>
#include<conio.h>
void main()
{
        int gd = DETECT, gm, i;
        float x, y, x1, y1, x2, y2, dx, dy, step;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        cout << "Enter the value of x1 and y1:";
        cin >> x1 >> y1;
        cout << "Enter the value of x2 and y2:";
        cin >> x2 >> y2;
        dx = abs(x2 - x1);
        dy = abs(y2 - y1);
        if(dx >= dy)
                step = dx;
        else
                step = dy;
                dx = dx / step;
                dy = dy / step;
                x = x1;
                y = y1;
                i = 1;
        while(i <= step)
        {
                putpixel(x, y, 5);
                x = x + dx;
                y = y + dy;
```

```
                i = i + 1;
        }
        getch();
        closegraph();
}
```

Output:-

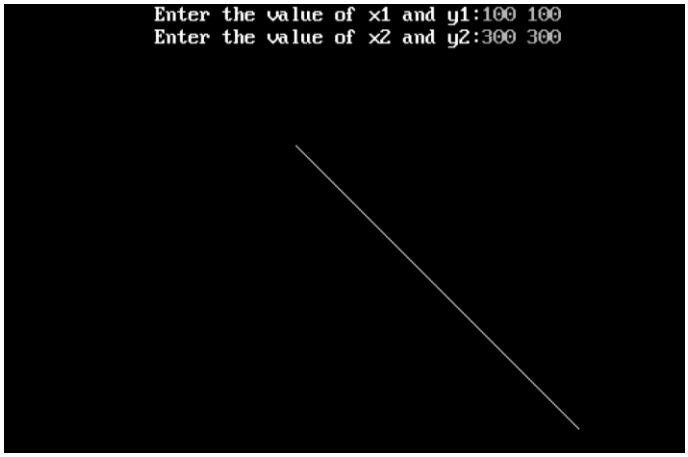


b) Develop the program for Bresenham's line drawing algorithm.

```
/* Bresenham's Line Drawing algorithm */
#include<graphics.h>
#include<iostream.h>
#include<math.h>
#include<conio.h>
void main()
{
        int gd = DETECT, gm, i;
        float x, y, x1, y1, x2, y2, dx, dy, e;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        cout << "Enter the value of x1 and y1:";
        cin >> x1 >> y1;
        cout << "Enter the value of x2 and y2:";
        cin >> x2 >> y2;

        dx = abs(x2 - x1); /* calculate dx and dy*/
        dy = abs(y2 - y1);

        x = x1; /*initialize x and y & error term*/
        y = y1;
        e = (dy/dx) - 0.5;
        putpixel(x, y, 15);

        for(i=1; i<=dx; i++) /* Obtain next pixels and paints*/
        {
                while(e>0)
                {
                        y = y + 1;
                        e = e - 1;
                }
```

```
                x = x + 1;
                e = dy/dx + e;
                putpixel(x, y, 15);
        }
        getch();
        closegraph();

}
```

Output:-



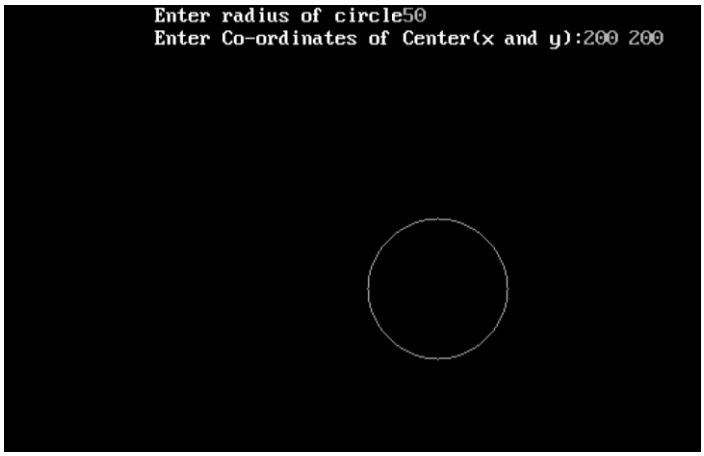
Practical 5

a) Develop the program for the Mid-Point circle drawing algorithm.

```
/* Mid-Point Circle Drawing Algorithm */
#include<graphics.h>
#include<iostream.h>
#include<conio.h>
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 + y, y0 - x, 7);
```

```
                putpixel(x0 + x, y0 - y, 7);

                if(err <= 0)
                {
                        y += 1;
                        err += 2*y + 1;
                }

                if(err > 0)
                {
                        x -= 1;
                        err -= 2*x + 1;
                }
        }
}
int main()
{
        int gd = DETECT, gm, error, x, y, r;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");

        cout<< "Enter radius of circle";
        cin >> r;

        cout << "Enter Co-ordinates of Center(x and y):";
        cin >> x >> y;
        drawcircle(x, y, r);
        getch();
        closegraph();
        return 0;
}
```

Output:-

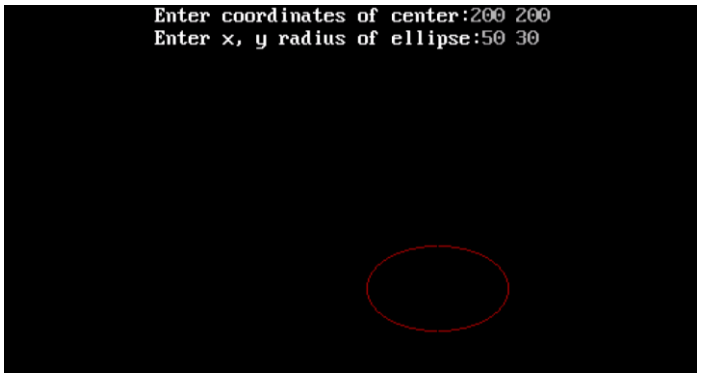


b) Develop the program for the Mid-Point Ellipse drawing algorithm.

```
/* Mid-Point Ellipse Drawing Algorithm */
#include<graphics.h>
#include<stdlib.h>
#include<iostream.h>
#include<conio.h>
void main()
{
        clrscr();
        int gd = DETECT, gm;
        int xc, yc, x, y;
        float p;
        long rx, ry;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");

        cout<< "Enter coordinates of center:";
        cin >> xc >> yc;

        cout << "Enter x, y radius of ellipse:";
        cin >> rx >> ry;

        //Region 1
        p = ry * ry - rx * rx * ry + rx * rx / 4;
        x = 0;
        y = ry;
        while(2.0 * ry * ry * x <= 2.0 * rx * rx * y)
        {
                if(p < 0)
                {
                        x++;
                        p = p + 2 * ry * ry * x + ry * ry;
                }

                else
                {
                        x++;
                        y--;
                        p = p + 2 * ry * ry * x - 2 * rx * rx * y - ry *ry;
                }
                putpixel(xc + x, yc + y, RED);
                putpixel(xc + x, yc - y, RED);
                putpixel(xc - x, yc + y, RED);
                putpixel(xc - x, yc - y, RED);
        }

        //Region 2
```

```
        p = ry * ry * (x + 0.5)*(x + 0.5) + rx * rx * (y - 1)*(y - 1) - rx * rx * ry * ry;
        while(y > 0)
        {
                if(p <= 0)
                {
                        x++;
                        y --;
                        p = p + 2 * ry * ry * x - 2 * rx * rx * y + rx * rx;
                }
                else
                {
                        y--;
                        p = p - 2 * rx * rx * y + rx * rx;
                }
                putpixel(xc + x, yc + y, RED);
                putpixel(xc + x, yc - y, RED);
                putpixel(xc - x, yc + y, RED);
                putpixel(xc - x, yc - y, RED);
        }
        getch();
        closegraph();
}
```

Output:-



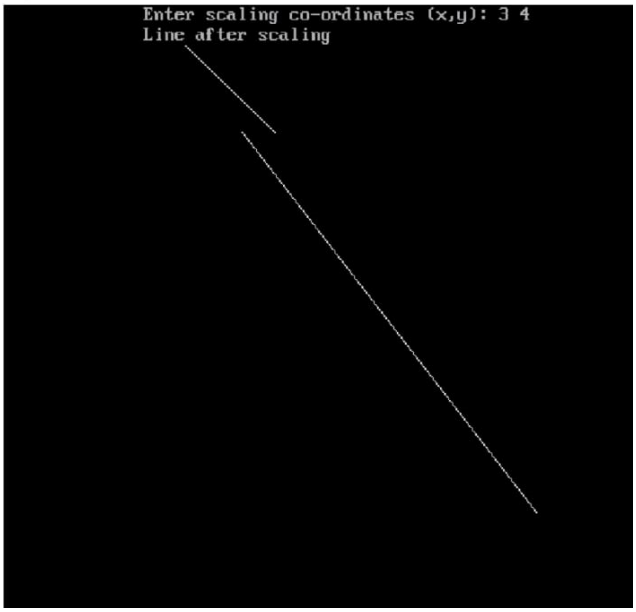
Practical 6

a) Write a program to implement 2D Scaling.

```
/* 2D Scaling */
#include<graphics.h>
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
```

```
#include<conio.h>
void main()
{
        int gd = DETECT, gm, errorcode, i;
        float x, y, x1, y1, x2, y2;

        printf("Enter the 2 lines end points");
        printf("\t x1, y1, x2, y2");
        scanf("%d%d%d%d", &x1, &y1, &x2, &y2);
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");

        line(x1,y1,x2,y2);
        printf("Enter scaling co-ordinates");
        printf("x, y");
        scanf("%d%d", &x, &y);

        x1 = (x1*x);
        y1 = (y1*y);
        x2 = (x2*x);
        y2 = (y2*y);
        printf("Line after Scaling");
        line(x1,y1,x2,y2);
        getch();
        closegraph();
}
```

Output:-



b) Write a program to perform 2D Translation.

```
/* 2D Translation */
#include<graphics.h>
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<conio.h>
void main()
{
        int gd = DETECT, gm, errorcode, i;
        float x, y, x1, y1, x2, y2;

        printf("Enter the 2 lines end points");
        printf("x1, y1, x2, y2");
        scanf("%d %d %d %d", &x1, &y1, &x2, &y2);
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");

        line(x1, y1, x2, y2);
        scanf("Enter translation co-ordinates");
        printf("x, y");
        scanf("%d %d", &x, &y);

        x1 = (x1 + x);
        y1 = (y1 + y);
        x2 = (x2 + x);
        y2 = (y2 + y);
        printf("Line after translation");
        line(x1, y1, x2, y2);
        getch();
        closegraph();
}
```

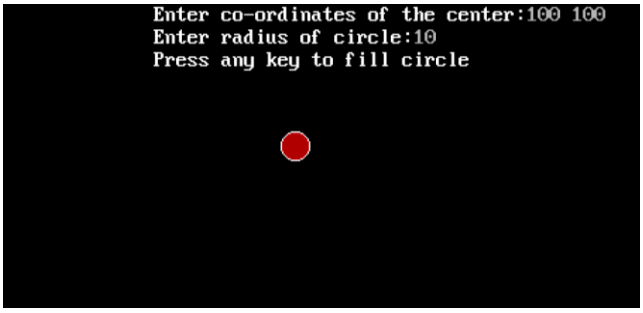
Output:-

Practical 9

a) **Write a program to fill a circle using Flood Fill Algorithm**.

```
/* Flood Fill Algorithm */
#include<graphics.h>
#include<dos.h>
#include<iostream.h>
#include<conio.h>
void ffill(int x, int y, int o_col, int n_col)
{
        int current = getpixel(x, y);
        if(current == o_col)
        {
                delay(1);
                putpixel(x, y, n_col);
                ffill(x+1, y, o_col, n_col);
                ffill(x-1, y, o_col, n_col);
                ffill(x, y+1, o_col, n_col);
                ffill(x, y-1, o_col, n_col);
        }
}
void main()
{
        int xc, yc, r;
        int gd = DETECT, gm;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        cout << "Enter co-ordinates of the center:";
        cin >> xc >> yc;
        cout << "Enter radius of circle:";
        cin >> r;
        circle(xc, yc, r);
        cout << "Press any key to fill circle";
        getch();
        ffill(xc, yc, BLACK, RED);
        getch();
        closegraph();
}
```

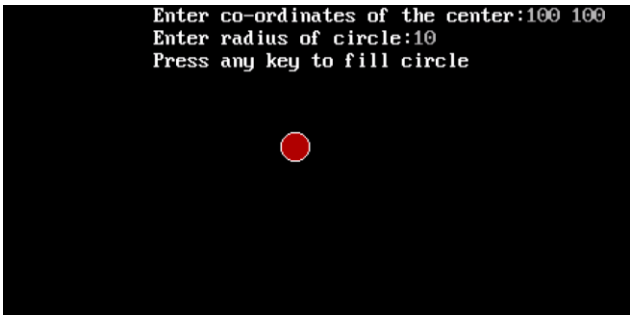
Output:-



b) Write a program to fill a circle using Boundary Fill Algorithm.

```
/* Boundary Fill Algorithm */
#include<graphics.h>
#include<dos.h>
#include<iostream.h>
#include<conio.h>
void bfill(int x, int y, int f_col, int b_col)
{
        int current = getpixel(x, y);
        if(current != f_col && current != b_col)
        {
                delay(1);
                putpixel(x, y, f_col);
                bfill(x+1, y, f_col, b_col);
                bfill(x-1, y, f_col, b_col);
                bfill(x, y+1, f_col, b_col);
                bfill(x, y-1, f_col, b_col);
        }
}
void main()
{
        int xc, yc, r;
        int gd = DETECT, gm;
        initgraph(&gd, &gm, "c:\\turboc3\\bgi");
        cout << "Enter co-ordinates of the center:";
        cin >> xc >> yc;
        cout << "Enter radius of circle:";
        cin >> r;
        circle(xc, yc, r);
        cout << "Press any key to fill circle";
        getch();
        bfill(xc, yc, RED, WHITE);
        getch();
        closegraph();
}
```

Output:-

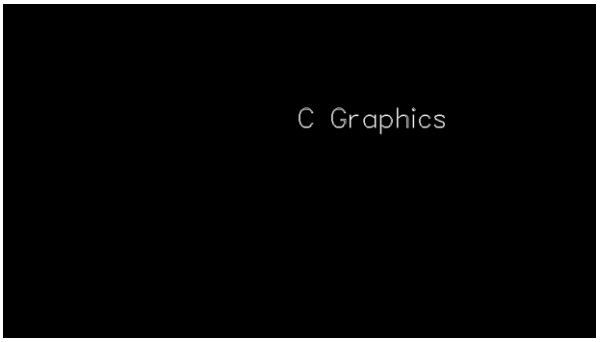


Practical 10

a) Develop a simple text screen saver using graphics functions.

```
#include<graphics.h>
#include<stdio.h>
#include<conio.h>
void main()
{
        int gd = DETECT, gm, i, maxx, maxy, key = 0;
        initgraph(&gd, &gm, "c://turboc3//bgi");
        maxx = getmaxx();
        maxy = getmaxy();
        while(!kbhit()) /* used to determine if a key has been pressed or not */
        {
                for(i=0; i<maxy; i++)
                {
                        cleardevice();
                        settextstyle(3, 0, 5); //font direction size
                        outtextxy(maxx/2, i, "C Graphics");
                        delay(30);
                }
        }
        getch();
}
```

Output:-

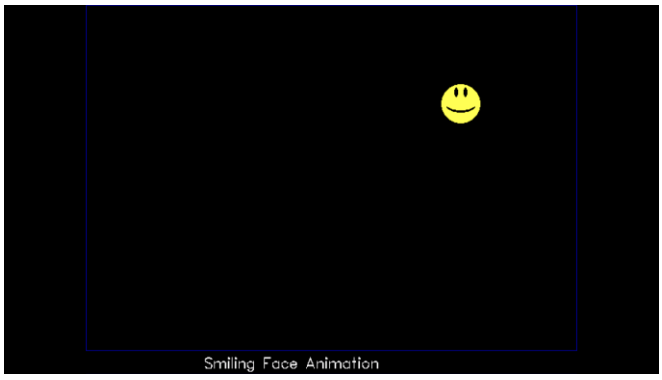


b) Perform smiling face animation using graphics functions.

```
/* Smiling Face Animation */
#include<graphics.h>
#include<conio.h>
#include<stdlib.h>

main()
{
  int gd = DETECT, gm, area, temp1, temp2, left = 25, top = 75;
  void *p; /* pointer to the bitmap image in memory */

  initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");

  setcolor(YELLOW);
  circle(50, 100, 25); /* x, y, radius */
  setfillstyle(SOLID_FILL, YELLOW);
  floodfill(50, 100, YELLOW); /* x, y */

  setcolor(BLACK);
  setfillstyle(SOLID_FILL, BLACK);
  fillellipse(44, 85, 2, 6); /* x, y, xradius, yradius */
  fillellipse(56, 85, 2, 6); /* eye */

  ellipse(50, 100, 205, 335, 20, 9); /* x, y, stangle, endangle, xradius, yradius */
  ellipse(50, 100, 205, 335, 20, 10); /* mouth */
  ellipse(50, 100, 205, 335, 20, 11); /* nose */

  area = imagesize(left, top, left + 50, top + 50); /*right , bottom */
  p = malloc(area);

  setcolor(WHITE);
  settextstyle(SANS_SERIF_FONT, HORIZ_DIR, 2); /* char size */
  outtextxy(155, 451, "Smiling Face Animation"); /* x, y */
```

```
  setcolor(BLUE);
  rectangle(0, 0, 639, 449);

  while(!kbhit())
  {
    temp1 = 1 + random (588);
    temp2 = 1 + random (380);

    getimage(left, top, left + 50, top + 50, p);
    putimage(left, top, p, XOR_PUT);
    putimage(temp1 , temp2, p, XOR_PUT);
    delay(100);
    left = temp1;
    top = temp2;
  }

  getch();
  closegraph();
  return 0;
}
```

Output:-



c) Draw the moving car on the screen.

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

main()
{
  int i, j = 0, gd = DETECT, gm;

  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

  settextstyle(DEFAULT_FONT,HORIZ_DIR,2);
```

```
  outtextxy(25,240,"Press any key to view the moving car");

  getch();
  setviewport(0,0,639,440,1);

  for (i = 0; i <= 420; i = i + 10, j++)
  {
    rectangle(50+i,275,150+i,400);
    rectangle(150+i,350,200+i,400);
    circle(75+i,410,10);
    circle(175+i,410,10);
    setcolor(j);
    delay(100);

    if (i == 420)
      break;

    clearviewport();
  }

  getch();
  closegraph();
  return 0;
}
```

Output:-