Lab - 4

/*Bresenham circle drawing*/

Pseudocode:

Draw the circle for a given radius 'r' and centre (x_c, y_c) starting from (0, r) and move in first quadrant till x=y (i.e. 45 degree) ,first octant.

Initial conditions :

- x = 0
- y = r
- p = 3 - (2 * r)

Steps:

- **Step 1** : Set initial values of (x_c, y_c) and (x, y)
- **Step 2** : Calculate decision parameter p = 3 – (2 * r).
- **Step 3** : call displayBresenhmCircle(int x_c, int y_c, int x, int y) method to display initial(0,r) point.
- **Step 4** : Repeat steps 5 to 8 until x < = y
- **Step 5** : Increment value of x.
- **Step 6** : If p < 0, set p = p + (4*x) + 6
- **Step 7** : Else, set p = p + 4 * (x – y) + 10 and decrement y by 1.
- **Step 8** : call displayBresenhmCircle(int x_c, int y_c, int x, int y) method.
- **Step 9** : Exit.

C-Program to demonstrate Bresenham Circle drawing algorithm.

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>


void Drawcircle(int, int, int);

void symmetry(int, int, int, int);
```

```
void main() {
  int gd = DETECT, gm;
  int x, y, r;
 initgraph(&gd, &gm, "C:\\TurboC3\\BGI");

   printf("Enter The center of circle: ");
   scanf("%d%d",&x,&y);

   printf("Enter the radius of the circle:\n");
   scanf("%d",&r);

  Drawcircle(x, y, r);
  getch();
  closegraph();
}

void Drawcircle(int x1, int y1, int r) {
  int x = 0, y = r, p = 3 - 2 * r;

  symmetry(x1, y1, x, y);
  while (x < y) {
    x++;
    if (p < 0)
```

```
            p += 4 * x + 6;
        else {

            p += 4 * (x - y) + 10;
          y--;

        }
        symmetry(x1, y1, x, y);
    }
}

void symmetry(int xctr, int yctr, int x, int y) {
    putpixel(xctr + x, yctr + y, 1);
    putpixel(xctr - x, yctr + y, 1);
    putpixel(xctr + x, yctr - y, 1);
    putpixel(xctr - x, yctr - y, 1);
    putpixel(xctr + y, yctr + x, 1);
    putpixel(xctr - y, yctr + x, 1);
    putpixel(xctr + y, yctr - x, 1);
    putpixel(xctr - y, yctr - x, 1);
    getch();
}
```

Lab -5

/* MID-POINT CIRCLE DRAWING */

Pseudocode:

Draw the circle for a given radius 'r' and centre (x_c, y_c) starting from (0, r) and move in first quadrant till x=y (i.e. 45 degree) ,first octant.

Initial conditions :

- x = 0
- y = r
- p = 1 - r

Steps:

- **Step 1** : Set initial values of (x_c, y_c) and (x, y)
- **Step 2** : Calculate decision parameter p = 1 - r.
- **Step 3** : call displayMidPointCircle(int x_c, int y_c, int x, int y) method to display initial(0,r) point.
- **Step 4** : Repeat steps 5 to 8 until x < = y
- **Step 5** : Increment value of x.
- **Step 6** : If p < 0, set p = p + (2*x) + 3
- **Step 7** : Else, set p = p + 2 * (x – y) + 5 and decrement y by 1.
- **Step 8** : call displayMidPointCircle(int x_c, int y_c, int x, int y) method.
- **Step 9** : Exit.

C-Program to demonstrate mid-point circle drawing algorithm

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>


void main() {

  int gd = DETECT, gm;

  int x, y, r;
```

```
  void Drawcircle(int, int, int);
   printf("Enter The center of circle: ");
   scanf("%d%d",&x,&y);

   printf("Enter the radius of the circle:\n");
   scanf("%d",&r);

  initgraph(&gd, &gm, "C:\\TurboC3\\BGI");
  Drawcircle(x, y, r);
  getch();
  closegraph();
}

void Drawcircle(int x1, int y1, int r) {
  int x = 0, y = r, p = 1 - r;
  void cliplot(int, int, int, int);
  cliplot(x1, y1, x, y);
  while (x < y) {
    x++;
    if (p < 0)
          p += 2 * x + 3;
    else {
      y--;
          p += 2 * (x - y) + 5;
    }
```

```
      cliplot(x1, y1, x, y);
    }
}

void cliplot(int xctr, int yctr, int x, int y) {
    putpixel(xctr + x, yctr + y, 1);
    putpixel(xctr - x, yctr + y, 1);
    putpixel(xctr + x, yctr - y, 1);
    putpixel(xctr - x, yctr - y, 1);
    putpixel(xctr + y, yctr + x, 1);
    putpixel(xctr - y, yctr + x, 1);
    putpixel(xctr + y, yctr - x, 1);
    putpixel(xctr - y, yctr - x, 1);
    getch();
}
```