Name- Kanika Bisht
Course- BCA Sem- VI Sec- C
Roll. No- 1121071
Subject → Computer Graphics lab End Sem

Ans 1. Bresenham Line Drawing Algorithm

Algorithm :-

Step 1:- Start

~~Declare Input Declare~~

Step 2: Declare x, y, x1, y1, x2, y2, dx, dy, steps, p in float data type.

Step 3: Declare gm and initialize gd = Detect & i=1.

Step 4: Enter coordinates x1 & y1 of first point.

Step 5: Enter coordinates x2 & y2 of second point.

Step 6: Initialize graph by using initgraph (&gd, &gm, " ");

Step 7: Calculate, dx = x2 - x1,
dy = y2 - y1,
steps = dx - 1.

Step 8: Initialize decision parameter

$pk = (2 * dy) - dx$

Step 9: Initialize $p = pk$, $x = x1$, $y = y1$.

Step 10: Repeat step 11 to step 13 while $i <= steps$

Step 11: Check if $p < 0$, then

putpixel (x, y, BLUE);
x = x + 1;
y = y;
p = p + (2 * dy);

otherwise go to step 12.

Step 12: putpixel (x, y, BLUE);
x = x + 1;
y = y + 1;
p = p + (2 * dy) - (2 * dx);

Step 13: Increment i by one.

Step 14: Close the graph

Step 15. Stop

Ranjika

Code :-

```
#include <stdio.h>
#include <graphics.h>
void main()
{
    float x, y, x1, y1, x2, y2, dx, dy, steps, p;
    int i=1, gd=DETECT, gm;
    printf("Enter (x1, y1): ");
    scanf("%f %f", &x1, &y1);
    printf("Enter (x2, y2): ");
    scanf("%f %f", &x2, &y2);
    initgraph(&gd, &gm, "");
    dx = x2 - x1;
    dy = y2 - y1;
    steps = dx - 1;
    int pk = (2*dy) - dx;
    p = pk;
    x = x1;
    y = y1;
    while (i <= steps)
    {
        if (p<0)
        {
            putpixel(x, y, BLUE);
            x = x+1; y = y; p = p + (2*dy);
            delay(50);
        }
```

```
else
{
    putpixel (x, y, BLUE);
    x = x+1;
    y = y+1;
    p = p+(2*dy) - (2*dx);
    delay (50);
}
i++;
}
getch();
closegraph();
}
```

Ravika

Ans-2. Mide Point Circle Drawing Algorithm

Algorithm:

Step1: Start

Step2: Initialize gd= DETECT & declare gm;

Step3: Declare r, x, y, p & initialize centre coordinates xc=200 & yc=200.

Step4: Enter radius r.

Step5: Initialize the graph

Step6: Initialize x=0, y=r & calculate

p = 1-r.

~~Step7: Repeat steps from step 8.~~

Step7: Check FOR condition;

for(x=0; x<=y; x++)

Step8: If condition is true, then go to step 9, otherwise goto step 12.

Step9: Check whether p<0, then

y=y;

p=p+(2*x)+1

otherwise

y=y+1;

p=p+(2*x)-(2*y)+1;

Step 10; Plot pixels using putpixel function

```
putpixel (xc+x, yc+y, 7);
putpixel (xc+y, yc+x, 7);
putpixel (xc-x, yc+y, 7);
putpixel (xc-y, yc+x, 7);
putpixel (xc-x, yc-y, 7);
putpixel (xc-y, yc-x, 7);
putpixel (xc+x, yc-y, 7);
putpixel (xc+y, yc-x, 7);
```

Step 11: Go to step 7

Step 12: Close the graph

Step 13: Stop

Ranika

Code :-

```
#include <stdio.h>
#include <graphics.h>
int main()
{
  int gd = DETECT, gm;
  int r, x, y, p, xc = 200, yc = 200;
  printf("Enter radius :");
  scanf("%d", &r);
  initgraph(&gd, &gm, " ");
  x = 0;
  y = r;
  p = 1 - r;
  for(x=0; x<=y; x++)
  {
      if(p<0)
      {
          y = y;
          p = p + (2*x) + 1;
      }
      else
      {
          y = y - 1;
          p = p + (2*x) - (2*y) + 1;
      }
```

```
        putpixel (xc+x, yc+y, 7);
        putpixel (xc+y, yc+x, 7);
        putpixel (xc-x, yc+y, 7);
        putpixel (xc-y, yc+x, 7);
        putpixel (xc-x, yc-y, 7);
        putpixel (xc-y, yc-x, 7);
        putpixel (xc+x, yc-y, 7);
        putpixel (xc+y, yc-x, 7);
    }
    getch();
    closegraph();
    return 0;
}
```

Ranika