Name : Sabarivasan V
Class : CSE - B
Reg No : 205001085

1. Using Bresenham's Line drawing and Circle drawing algorithms draw a ferris wheel as shown below.

```
#include <stdio.h>
#include <iostream>
#include <GLUT/GLUT.h>

using namespace std;

int pntX, pntY, r;

void myInit(void)
{
   glClearColor(1.0, 1.0, 1.0, 0.0);
   glColor3f(0.0f, 0.0f, 0.0f);
   glPointSize(4.0);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(-320.0, 320.0, -240.0, 240.0);
}

void plot(int x, int y,int pntx,int pnty)
{
   glBegin(GL_POINTS);
   glVertex2i(x + pntx, y + pnty);
   glEnd();
}
```

```
void midPointCircleAlgo(int pntx, int pnty, int radius)
{

   int x = 0;
   int y = radius;
   float decision = 5 / 4 - r;
   plot(x, y,pntx,pnty);

   while (y > x)
   {
      if (decision < 0)
      {
         x++;
         decision += 2 * x + 1;
      }
      else
      {
         y--;
         x++;
         decision += 2 * (x - y) + 1;
      }

      plot(x, y,pntx,pnty);
      plot(x, -y,pntx,pnty);
      plot(-x, y,pntx,pnty);
      plot(-x, -y,pntx,pnty);
      plot(y, x,pntx,pnty);
      plot(-y, x,pntx,pnty);
      plot(y, -x,pntx,pnty);
      plot(-y, -x,pntx,pnty);
   }
}

void draw_pixel(int x, int y)
{
   glBegin(GL_POINTS);
   glVertex2i(x, y);
   glEnd();
}

void draw_line(int xstart, int ystart, int xend,  int yend)
{
   int dx, dy, i, e;
   int incx, incy, inc1, inc2;
```

```
int x, y;
dx = abs(xend - xstart);
dy = abs(yend - ystart);
incx = 1;
if (xend < xstart)
   incx = -1;
incy = 1;
if (yend < ystart)
   incy = -1;
x = xstart;
y = ystart;
if (dx > dy)
{
   draw_pixel(x, y);
   e = 2 * dy - dx;
   inc1 = 2 * (dy - dx);
   inc2 = 2 * dy;
   for (i = 0; i < dx; i++)
   {
      if (e >= 0)
      {
         y += incy;
         e += inc1;
      }
      else
         e += inc2;
      x += incx;
      draw_pixel(x, y);
   }
}
else
{
   draw_pixel(x, y);
   e = 2 * dx - dy;
   inc1 = 2 * (dx - dy);
   inc2 = 2 * dx;
   for (i = 0; i < dy; i++)
   {
      if (e >= 0)
      {
         x += incx;
         e += inc1;
      }
      else
```

```
            e += inc2;
        y += incy;
        draw_pixel(x, y);
      }
   }
}

void myDisplay(void)
{
   glClear(GL_COLOR_BUFFER_BIT);
   glColor3f(0.0, 0.0, 0.0);
   glPointSize(1.0);

   midPointCircleAlgo(0,0,150);
   midPointCircleAlgo(0,0,20);
   midPointCircleAlgo(0,0,130);

   draw_line(0,0,-200,-200);
   draw_line(0,0, 200, -200);

   midPointCircleAlgo(0,145,20);
   midPointCircleAlgo(0,-145,20);
   midPointCircleAlgo(145,0,20);
   midPointCircleAlgo(-145,0,20);
   midPointCircleAlgo(100,100,20);
   midPointCircleAlgo(-100,-100,20);
   midPointCircleAlgo(100,-100,20);
   midPointCircleAlgo(-100,100,20);

   glFlush();
}


int main(int argc, char **argv)

{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize(640, 480);
   glutInitWindowPosition(100, 150);
   glutCreateWindow("ferris wheel ");
   glutDisplayFunc(myDisplay);
   myInit();
   glutMainLoop();
```

```
    return 0;
}
```

Output :