```
/* Problem 3--Unfair Slicing
   This required a little bit of trigonometry and algebra.  The idea is
   that these weird shapes can be divided into a triangle and a circular
   cap.  A triangle's area can always be found with Heron's Formula, and
   a circular cap is never "skewed" and can be found by subtracting the
   triangle from a slice centered at the true center containing the same
   cap. */
import java.io.*;
import java.util.*;

public class prob3 {

 private static Scanner in;
 private static PrintWriter out;
 private static int cs;
 private static double x0, y0, r;
 private static int slices;

 public static void main (String[] args) throws Exception {

  in = new Scanner (new File ("prob3.in"));
  out = new PrintWriter ("prob3.out");
  cs = 1;
  while (true) {
   x0 = in.nextDouble(); y0 = in.nextDouble(); r = in.nextDouble();
   slices = in.nextInt();
   if (slices==0) break;
   Process ();
  }
  in.close ();
  out.close ();
 }

 public static void Process () throws Exception {

  out.printf ("Case %d: The areas are:\r\n",cs++);
  for (int i=0; i < slices; i++) {/*Finding the angles in (x,y) space*/
   double firstang = Math.PI/2 - i*2*Math.PI/slices;
   double secondang = Math.PI/2 - (i+1)*2*Math.PI/slices;
   double xk1 = Math.cos (firstang); /* The "direction" of the two */
   double yk1 = Math.sin (firstang); /* angles */
   double xk2 = Math.cos (secondang);
   double yk2 = Math.sin (secondang); /* Where do the slices intersect*/
   double[] ip = IntersectionPoint (xk1,yk1); /* the circle? */
   double p1x = ip[0];
   double p1y = ip[1];
   ip = IntersectionPoint (xk2,yk2);
   double p2x = ip[0];
   double p2y = ip[1];
   double area = computeArea (p1x,p1y,p2x,p2y); /* grab area */
   out.printf ("%.1f\r\n",area);
  }
  out.printf ("\r\n");
 }

 public static double computeArea (double p1x,
                            double p1y, double p2x, double p2y)
                            throws Exception {

   /* Find lengths of three sides of triangle part of slice */
  double s1 = Math.sqrt ((p1x-x0)*(p1x-x0)+(p1y-y0)*(p1y-y0));
  double s2 = Math.sqrt ((p2x-x0)*(p2x-x0)+(p2y-y0)*(p2y-y0));
  double s3 = Math.sqrt ((p2x-p1x)*(p2x-p1x)+(p2y-p1y)*(p2y-p1y));
  double s = (s1+s2+s3)/2; /* Apply Heron's Formula */
  double triarea = Math.sqrt (s*(s-s1)*(s-s2)*(s-s3));
  double angle = Math.acos (1-s3*s3/(2*r*r)); /* Central angle of cap */
  double straightslice = angle/2 *r*r; /* area of slice centered at
                                          center */
  double straighttriangle = 0.5*r*r*Math.sin(angle); /*Triangular part*/
  double arc = straightslice-straighttriangle; /* area of cap */
  double weirdslice = triarea+arc; /* Area of strange shape */
  return weirdslice;
 }

 public static double[] IntersectionPoint (double xk, double yk)
      throws Exception {

   /* set up and solve quadratic to find intersection point */
  double a = xk*xk+yk*yk;
  double b = 2*(x0*xk+y0*yk);
  double c = x0*x0+y0*y0-r*r;
  double t = (-b+Math.sqrt(b*b-4*a*c))/(2*a);
  double[] ip = new double[2];
  ip[0] = x0+xk*t; ip[1] = y0+yk*t;
  return ip;
 }


}
```