```
 1  /* Problem 6--Sphere Reflection
 2     This is not too bad if you know linear algebra.  You don't actually
 3     need calculus for this.  Draw a vector from the center of the sphere.
 4     Project each vector onto the surface of the sphere and find the
 5     average, and project that onto the surface.  That's the reflection
 6     point.  */
 7
 8  import java.io.*;
 9  import java.util.*;
10
11  public class prob6 {
12
13   private static Scanner in;
14   private static PrintWriter out;
15   private static int cs;
16   private static double x, y, z, r, sx, sy, sz, ex, ey, ez;
17
18   public static void main (String[] args) throws Exception {
19
20    in = new Scanner (new File ("prob6.in"));
21    out = new PrintWriter ("prob6.out");
22    cs = 1;
23    while (true) { //Read in the 10 numbers
24     x = in.nextDouble ();y = in.nextDouble ();z = in.nextDouble();
25     r = in.nextDouble ();sx = in.nextDouble(); sy = in.nextDouble();
26     sz = in.nextDouble(); ex = in.nextDouble(); ey = in.nextDouble();
27     ez = in.nextDouble();
28     if (x==0 && y==0 && z==0 && r==0 && sx==0 && sy==0 && sz==0 && ex==0
29         && ey==0 && ez==0) break;
30     Process (); //Process the data case
31    }
32    in.close ();
33    out.close ();
34   }
35
36   //Process the data case
37   public static void Process () throws Exception {
38
39    if (Inside()) { //One or both particles are inside the sphere
40     out.print ("Case "+(cs++)+": Cannot aim\r\n\r\n");
41     return;
42    }  //Vectors to points
43    double v1x = sx-x, v1y = sy-y, v1z = sz-z;
44    double v2x = ex-x, v2y = ey-y, v2z = ez-z;
45    double v1n = Math.sqrt (v1x*v1x + v1y*v1y + v1z*v1z); //Length
46    double v2n = Math.sqrt (v2x*v2x + v2y*v2y + v2z*v2z);
47       //Projection onto surface of sphere
48    double p1x = x+v1x/v1n*r, p1y = y+v1y/v1n*r, p1z = z+v1z/v1n*r;
49    double p2x = x+v2x/v2n*r, p2y = y+v2y/v2n*r, p2z = z+v2z/v2n*r;
50    double newx = (p1x+p2x)/2, newy = (p1y+p2y)/2, newz = (p1z+p2z)/2;
51     //halfway between
52    double vnx = newx-x, vny = newy-y, vnz = newz-z;//vector to halfway pt
53    double newn =Math.sqrt (vnx*vnx+vny*vny+vnz*vnz);//Normalized vector
54    if (Math.abs(newn) < 1e-5) { //Average is center of sphere, meaning
55                                 //they are on opposite sides
56     out.print ("Case "+(cs++)+": Cannot aim\r\n\r\n");
57     return;
58    }
59    double fx = x+vnx/newn*r, fy = y+vny/newn*r, fz = z+vnz/newn*r;
60    double maxangle = Math.acos (r/v1n);
61                //Angle from particle to tangent line
62    double s3 = //Find angle from particle to ricochet point
63       Math.sqrt ((sx-fx)*(sx-fx)+(sy-fy)*(sy-fy)+(sz-fz)*(sz-fz));
64    double angle = Math.acos((r*r+v1n*v1n-s3*s3)/(2*v1n*r));
```

```
65    if (angle > maxangle) { //If the ricochet point is too far away, the
66     out.print ("Case "+(cs++)+": Cannot aim\r\n\r\n");//sphere is between
67     return;               //the points
68    }
69    out.print ("Case "+(cs++)+": Aim for "); //Print point
70    out.printf ("(%.1f,%.1f,%.1f)\r\n\r\n",fx,fy,fz);
71   }
72
73   //Returns a boolean whether one or both points are inside the sphere
74   public static boolean Inside () throws Exception {
75
76    if ((sx-x)*(sx-x)+(sy-y)*(sy-y)+(sz-z)*(sz-z) <= r*r) return true;
77    if ((ex-x)*(ex-x)+(ey-y)*(ey-y)+(ez-z)*(ez-z) <= r*r) return true;
78    return false;
79   }
80
81  }
82
```