#include <cstdio>

#include <cmath>

using namespace std;

#define INF 1e9

#define EPS 1e-9

#define PI acos(-1.0)

double DEG\_to\_RAD(double d) { return d \* PI / 180.0; }

double RAD\_to\_DEG(double r) { return r \* 180.0 / PI; }

struct point\_i { int x, y; // whenever possible, work with point\_i

point\_i() { x = y = 0; } // default constructor

point\_i(int \_x, int \_y) : x(\_x), y(\_y) {} }; // constructor

struct point { double x, y; // only used if more precision is needed

point() { x = y = 0.0; } // default constructor

point(double \_x, double \_y) : x(\_x), y(\_y) {} }; // constructor

int insideCircle(point\_i p, point\_i c, int r) { // all integer version

int dx = p.x - c.x, dy = p.y - c.y;

int Euc = dx \* dx + dy \* dy, rSq = r \* r; // all integer

return Euc < rSq ? 0 : Euc == rSq ? 1 : 2; } //inside/border/outside

bool circle2PtsRad(point p1, point p2, double r, point &c) {

double d2 = (p1.x - p2.x) \* (p1.x - p2.x) +

(p1.y - p2.y) \* (p1.y - p2.y);

double det = r \* r / d2 - 0.25;

if (det < 0.0) return false;

double h = sqrt(det);

c.x = (p1.x + p2.x) \* 0.5 + (p1.y - p2.y) \* h;

c.y = (p1.y + p2.y) \* 0.5 + (p2.x - p1.x) \* h;

return true; } // to get the other center, reverse p1 and p2

int main() {

// circle equation, inside, border, outside

point\_i pt(2, 2);

int r = 7;

point\_i inside(8, 2);

printf("%d\n", insideCircle(inside, pt, r)); // 0-inside

point\_i border(9, 2);

printf("%d\n", insideCircle(border, pt, r)); // 1-at border

point\_i outside(10, 2);

printf("%d\n", insideCircle(outside, pt, r)); // 2-outside

double d = 2 \* r;

printf("Diameter = %.2lf\n", d);

double c = PI \* d;

printf("Circumference (Perimeter) = %.2lf\n", c);

double A = PI \* r \* r;

printf("Area of circle = %.2lf\n", A);

printf("Length of arc (central angle = 60 degrees) = %.2lf\n", 60.0 / 360.0 \* c);

printf("Length of chord (central angle = 60 degrees) = %.2lf\n", sqrt((2 \* r \* r) \* (1 - cos(DEG\_to\_RAD(60.0)))));

printf("Area of sector (central angle = 60 degrees) = %.2lf\n", 60.0 / 360.0 \* A);

point p1;

point p2(0.0, -1.0);

point ans;

circle2PtsRad(p1, p2, 2.0, ans);

printf("One of the center is (%.2lf, %.2lf)\n", ans.x, ans.y);

circle2PtsRad(p2, p1, 2.0, ans); // we simply reverse p1 with p2

printf("The other center is (%.2lf, %.2lf)\n", ans.x, ans.y);

return 0;

}