```cpp
// 幾何ライブラリ 2次元ベクトル内積/外積 線分距離 直線交点 凸包 円
#include <bits/stdc++.h>

using namespace std;

using ll = long long;
using vi = vector<int>;
using vll = vector<ll>;
using vvi = vector<vector<int>>;
using vvl = vector<vector<ll>>;

double eps = 0.0000001;
// asin(1.) * 2
double pi = 3.14159265358979323846264338327950288;
using p2 = complex<double>;
// x: real
// y: imag

double det(p2 v1, p2 v2) {
  return v1.real() * v2.imag() - v1.imag() * v2.real();
}

double dot(p2 v1, p2 v2) {
  return v1.real() * v2.real() + v1.imag() * v2.imag();
}

double dist2(p2 v) {
  return dot(v, v);
}

double dist(p2 v) {
  return sqrt(dist2(v));
}

bool same(double x, double y) { return abs(x - y) < eps; }

double dist2(p2 l1, p2 l2) { return dot(l1 - l2, l1 - l2); }

double dist(p2 l1, p2 l2) {
  return sqrt(dist2(l1, l2));
}

int ccw(p2 a, p2 b, p2 c) {
  b -= a;
  c -= a;
  if (det(b, c) > eps)return 1;
  if (det(b, c) < -eps) return -1;
  if (dot(b, c) < -eps) return 2;
  if (dist2(b) < dist2(c)) return -2;
  return 0;
```

```cpp
}

auto p2comp = [](const p2 &l, const p2 &r) {
  if (abs(l.real() - r.real()) > eps)
    return l.real() < r.real();
  return l.imag() < r.imag();
};

struct Line {
  p2 st, ed;

  Line(p2 st, p2 ed) : st(st), ed(ed) {}

  Line(double x1, double y1, double x2, double y2)
      : st(p2(x1, y1)), ed(p2(x2, y2)) {}

  Line(p2 st, double x, double y) : st(st), ed(p2(x, y)) {}

  Line(double x, double y, p2 ed) : st(p2(x, y)), ed(ed) {}

  double dist() { return sqrt(dist2(st, ed)); }

  bool isPalla(Line l) { return abs(det(ed - st, l.ed - l.st)) < eps; }

  double x() { return ed.real() - st.real(); }

  double y() { return ed.imag() - st.imag(); }

  p2 v() { return ed - st; }
};

// l1.st + (l1.st - l1.ed) * r.first = l2.st + (l2.st - l2.ed) * r.second
// 方程式を満たす(r.first, r.second)を返す
// l1.isPalla(l2) => (nan, nan)
pair<double, double> interP(Line l1, Line l2) {
  double a = l1.x();
  double b = -l2.x();
  double c = l1.y();
  double d = -l2.y();
  double inv = 1. / (a * d - c * b);
  double e1 = -l1.st.real() + l2.st.real();
  double e2 = -l1.st.imag() + l2.st.imag();
  return make_pair((d * e1 - b * e2) * inv, (-c * e1 + a * e2) * inv);
}

bool intersec(Line l1, Line l2) {
  if (l1.isPalla(l2))
    return false;
  auto r = interP(l1, l2);
  return eps < r.first && r.first < 1. - eps && eps < r.second &&
         r.second < 1. - eps;
}

double inter_r(Line l, p2 c) {
  p2 a = l.st, b = l.ed;
  return dot(a - c, l.v()) / dist2(l.v());
}

double dist(Line l, p2 c) {
  double r = inter_r(l, c);
  if (r < -eps) return dist(l.st, c);
  if (1. + eps < r) return dist(l.ed, c);
  return dist(l.st + l.v() * r, c);
}

p2 nearest(Line l, p2 c) {
  double r = inter_r(l, c);
  if (r < -eps) return l.st;
  if (1. + eps < r) return l.ed;
  return l.st + l.v() * r;
}

double dist(Line l1, Line l2) {
  return min(min(dist(l1, l2.st), dist(l1, l2.ed)), min(dist(l2, l1.st), dist(l2, l1.ed)));
}

struct Poly {
  vector<p2> ps;
  double d;

  Poly(vector<p2> ps) : ps(ps) {
    d = 0;
    for (int i = 0; i < ps.size(); i++) d += dist(ps[i], ps[(i + 1) % ps.size()]);
  }

  // 頂点上/辺上は微妙
  bool include(p2 p) {
    // 半直線
    Line l(p, p2(-10000, -1));
    int c = 0;
    for (int i = 0; i < ps.size(); i++) {
      if (intersec(l, Line(ps[i], ps[(i + 1) % ps.size()]))) c++;
    }
    return c % 2 == 1;
  }

  bool include(p2 p, bool on_vert, bool on_edge) {
    for (auto &q : ps) if (dist(p, q) < eps) return on_vert;
    for (int i = 0; i < ps.size(); i++) {
      if (ccw(ps[i], ps[(i + 1) % ps.size()], p) == 0) return on_edge;
    }
    return include(p);
  }

  bool intersecl(Line l) {
    for (int i = 0; i < ps.size(); i++) {
      if (intersec(l, Line(ps[i], ps[(i + 1) % ps.size()])))
        return true;
    }
    return false;
  }
};

struct Circle {
  p2 p;
  double r;

  Circle(p2 p, double r) : p(p), r(r) {}

  bool include(p2 l) { return dist2(p, l) < r * r + eps; }

  // 円同士の交点
  // 存在すれば2つ
  vector<p2> intersec(Circle c) {
    p2 diff = c.p - p;
    double dist = dot(diff, diff);
    double a = (dist + r * r - c.r * c.r) / 2.;
    double D = dist * r * r - a * a;
    if (D < eps)
      return vector<p2>();
    double Dsqrt = sqrt(D);
    vector<p2> ps;
    ps.emplace_back((a * diff.real() + diff.imag() * Dsqrt) / dist + p.real(),
                    (a * diff.imag() - diff.real() * Dsqrt) / dist + p.imag());
    ps.emplace_back((a * diff.real() - diff.imag() * Dsqrt) / dist + p.real(),
                    (a * diff.imag() + diff.real() * Dsqrt) / dist + p.imag());
    return ps;
  }
};

// 半時計回り
struct ConX {
  vector<p2> ps;

  // graham scan
  // ref: プログラミングコンテストチャレンジブック p233
  ConX(vector<p2> v) {
    sort(v.begin(), v.end(), p2comp);

    int k = 0, n = v.size();
    ps.resize(n * 2);
    for (int i = 0; i < n; i++) {
      while (k > 1 && det(ps[k - 1] - ps[k - 2], v[i] - ps[k - 1]) < eps)
        k--;
      ps[k++] = v[i];
    }
    for (int i = n - 2, t = k; i >= 0; i--) {
      while (k > t && det(ps[k - 1] - ps[k - 2], v[i] - ps[k - 1]) < eps)
        k--;
      ps[k++] = v[i];
    }
    ps.resize(k - 1);
  }

  Poly toPoly() {
    return Poly(ps);
  }

  size_t size() { return ps.size(); }
};

int n;
vector<double> rs;
vector<p2> ps;
vector<Circle> cs;

bool f(double l) {
  cs.clear();
  for (int i = 0; i < n; i++) {
    double rr = rs[i] * rs[i] - l * l;
    if (rr < eps)
      return false;
    cs.emplace_back(ps[i], sqrt(rr));
  }
  vector<p2> may;
  for (int i = 0; i < n; i++) {
    may.push_back(cs[i].p);
    for (int j = i + 1; j < n; j++) {
      auto v = cs[i].intersec(cs[j]);
      if (v.size() == 0)
        continue;
      may.push_back(v[0]);
      may.push_back(v[1]);
    }
  }
  for (auto &p : may) {
    bool ok = true;
    for (auto &c : cs) {
      if (!c.include(p)) {
        ok = false;
        break;
      }
    }
    if (ok)
      return true;
  }
  return false;
}

int main() {
  printf("%.20lf\n", asin(1.) * 2);
  cin.tie(nullptr);
  ios::sync_with_stdio(false);
  Poly p(vector<p2>({p2(0, 0), p2(1, 0.5), p2(2, 0), p2(2, 2), p2(1, 1.5), p2(0, 2)}));
  vector<double> xy({-1, 0, 1, 2, 3});
  for (auto &x : xy)
    for (auto &y : xy) {
      cerr << x << " " << y << " " << (p.include(p2(x, y), true, true) ? "YES" : "NO") << endl;
    }

  return 0;
}

```