Авторы:

Медведев Михаил Геннадиевич – кандидат физико-математических наук, доцент кафедры математической информатики факультета кибернетики Киевского национального университета имени Тараса Шевченко.

Присяжнюк Анатолий Васильевич – учитель-методист высшей категории специализированной школы с углубленным изучением информатики № 17 г. Бердичев Житомирской области.

Жуковский Сергей Станиславович – старший преподаватель кафедры прикладной математики и информатики Житомирского государственного университета имени Ивана Франко, учитель информатики городского лицея №25 имени Н. А. Щорса.

Задачи:

4, 839, 1043, 1501 – 1510, 2129 – 2133.

ОГЛАВЛЕНИЕ

ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

Основные определения. Простейшими фигурами на плоскости (геометрическими примитивами) являются ***точка*** и ***прямая***. ***Отрезком*** называется ограниченная часть прямой. ***Углом*** называется совокупность двух лучей, которые выходят из одной точки.

Вектором называется направленный отрезок прямой.

Пусть $a(x_1, y_1)$ и $b(x_2, y_2)$ – векторы.

Модуль вектора a: $|a| = \sqrt{x_1^2 + x_2^2}$;

Скалярное произведение векторов a и b: $(a, b) = |a| * |b| * \cos(a, b) = x_1 * x_2 + y_1 * y_2$.

Косое (псевдоскалярное) произведение векторов a и b ***на плоскости***:

$$a \times b = |a| * |b| * \sin(a, b) = \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} = x_1 y_2 - x_2 y_1 = - b \times a.$$

Косое произведение a и b положительно, если кратчайший поворот, который совмещает b с a, происходит по часовой стрелке, и отрицательно, если против.

Векторным произведением векторов a и b ***в пространстве*** называется вектор $a \times b$, определяемый следующими тремя условиями:

1. Модуль векторного произведения равен произведению модулей перемножаемых векторов на синус угла между ними: $|a \times b| = |a| * |b| * \sin(a, b)$;
2. Векторное произведение $a \times b$ перпендикулярно и вектору a и вектору b.
3. Упорядоченная тройка векторов a, b, $a \times b$ имеет положительную ориентацию.

Если координаты векторов равны $a(x_1, y_1, z_1)$ и $b(x_2, y_2, z_2)$, то

$$a \times b = \begin{vmatrix} i & j & k \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \end{vmatrix} = \left(\begin{vmatrix} y_1 & z_1 \\ y_2 & z_2 \end{vmatrix}, - \begin{vmatrix} x_1 & z_1 \\ x_2 & z_2 \end{vmatrix}, \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} \right)$$

Уравнение прямой на плоскости имеет вид $ax + by + c = 0$. Вектор нормали прямой имеет координаты (a, b), направляющий вектор прямой $(-b, a)$.

Прямая с угловым коэффициентом k, которая проходит через точку $(0, b)$, задается уравнением $y = kx + b$.

Уравнение прямой, проходящей через точки $A(x_1, y_1)$ и $B(x_2, y_2)$, имеет вид:

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$$

Доказательство. Прямой, проходящей через две точки A и B, является геометрическое место точек $X(x, y)$, для которых векторы AX и AB колинеарны. Векторы $AX(x - x_1, y - y_1)$ и $AB(x_2 - x_1, y_2 - y_1)$ колинеарны, если их координаты пропорциональны. Искомым уравнением прямой является условие пропорциональности координат векторов AX и AB.

Но в таком виде не представимы горизонтальные и вертикальные прямые, так как для них имеет место $y_1 = y_2$ или $x_1 = x_2$. ***Уравнение прямой*** можно переписать в виде:

$$(y_2 - y_1)\, x - (x_2 - x_1)\, y - x_1 y_2 + y_1 x_2 = 0$$

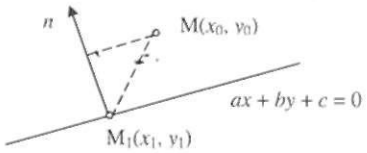
Расстояние между точками (x_1, y_1) и (x_2, y_2) равно $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

Расстояние от точки $M(x_0, y_0)$ **до прямой** $ax + by + c = 0$ равно

$$d = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

Доказательство. Пусть $n(a, b)$ – вектор нормали, проведенной к прямой из точки $M_1(x_1, y_1)$. $MA \perp n$, искомое расстояние $d = AM_1$. Обозначим через φ угол AM_1M. Тогда $d = M_1M * \cos\varphi$, скалярное произведение векторов n и M_1M (обозначим его через (n, M_1M)) равно $|n| * |M_1M| * \cos\varphi$, откуда

$$d = \frac{|(n, M_1M)|}{|n|} = \frac{|a(x_0 - x_1) + b(y_0 - y_1)|}{\sqrt{a^2 + b^2}} = \frac{|ax_0 + by_0 - ax_1 - by_1|}{\sqrt{a^2 + b^2}} = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$



Пересечение прямых. Пусть на плоскости заданы две прямые $a_1x + b_1y = c_1$ и $a_2x + b_2y = c_2$. Прямые параллельны, если их векторы нормалей параллельны, то есть если

$$\frac{a_1}{a_2} = \frac{b_1}{b_2}$$

В случае параллельности прямые совпадают, если

$$\frac{a_1}{a_2} = \frac{b_1}{b_2} = \frac{c_1}{c_2}$$

Если прямые не параллельны, то они пересекаются в одной точке. Точка их пересечения находится решением системы линейных уравнений

$$\begin{cases} a_1x + b_1y = c_1 \\ a_2x + b_2y = c_2 \end{cases}$$

методом Крамера:

$$d = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}, d_x = \begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix}, d_y = \begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}, x = \frac{d_x}{d}, y = \frac{d_y}{d}, d \neq 0$$

Функция kramer находит координаты пересечения двух прямых. На вход ей передаются значения a_1, b_1, c_1, a_2, b_2, c_2, а также адреса двух переменных x и y. Если система имеет единственное решение, то функция возвращает 0, а в переменных (x, y) возвращаются координаты точки пересечения. Если прямые параллельны и не имеют общих точек, функция возвращает 1. В случае совпадения прямых функция возвращает 2.

```
int kramer(double a1,double b1, double c1, double a2,double b2, double c2,
           double *x, double *y)
{
  double d = a1 * b2 - a2 * b1;
  double dx = c1 * b2 - c2 * b1;
  double dy = a1 * c2 - a2 * c1;
  if (!d) return (dx == 0.0) + 1;
  *x = dx/d; *y = dy/d;
  return 0;
}
```

Три точки $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$ лежат на одной прямой, если векторы $AB(x_2 - x_1, y_2 - y_1)$ и $AC(x_3 - x_1, y_3 - y_1)$ ***колинеарны***, то есть имеет место равенство

$$\frac{x_2 - x_1}{x_3 - x_1} = \frac{y_2 - y_1}{y_3 - y_1}$$

Для избежания деления на нуль и возможности использования формулы для любых входных данных, следует переписать равенство в виде $(x_2 - x_1) * (y_3 - y_1) = (x_3 - x_1) * (y_2 - y_1)$.

Расположение двух точек относительно прямой. Две точки A и B лежат по разные стороны от прямой CD, если векторные произведения $CD \times CA$ и $CD \times CB$ имеют разные знаки, то есть $(CD \times CA) * (CD \times CB) < 0$.

Если прямая задана уравнением $f(x, y) = ax + by + c = 0$, то две точки $A(x_1, y_1)$ и $B(x_2, y_2)$ лежат по разные от нее стороны тогда и только тогда, когда

$$f(x_1, y_1) * f(x_2, y_2) < 0$$

Направление поворота. Совершается движение от точки A до B, затем от B до C. При движении имеет место левый поворот (движение происходит против часовой стрелки), если $AB \times BC > 0$ и правый, если $AB \times BC < 0$.

Уравнение отрезка. Пусть $A(x_1, y_1)$, $B(x_2, y_2)$ – концы отрезка AB. Точка $X(x, y)$ принадлежит отрезку AB тогда и только тогда, когда $|AX| + |XB| = |AB|$ или в координатном виде:

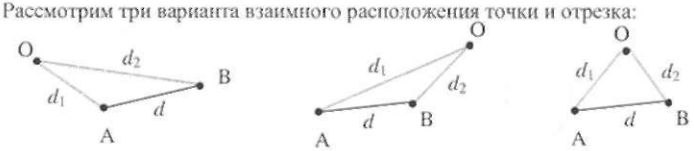
$$\sqrt{(x - x_1)^2 + (y - y_1)^2} + \sqrt{(x - x_2)^2 + (y - y_2)^2} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Параметрическое уравнение отрезка. Пусть $A(x_1, y_1)$, $B(x_2, y_2)$ – концы отрезка AB. Параметрическое уравнение отрезка имеет вид:

$$x(t) = x_1 + (x_2 - x_1) * t,\ y(t) = y_1 + (y_2 - y_1) * t,\ 0 \le t \le 1$$

Отрезок является геометрическим местом точек $(x(t), y(t))$, где $0 \le t \le 1$.

Расстояние от точки O(x, y) до отрезка AB, заданного координатами концов: $A(x_1, y_1)$, $B(x_2, y_2)$. Рассмотрим три варианта взаимного расположения точки и отрезка:



Обозначим $d = AB$, $d_1 = OA$, $d_2 = OB$. В первом случае угол A не острый, $d_2^2 \ge d_1^2 + d^2$, искомое расстояние равно d_1. Во втором случае угол B не острый, $d_1^2 \ge d_2^2 + d^2$, искомое расстояние равно d_2. Если углы A и B острые, то расстояние от точки O до отрезка AB равно длине высоты треугольника OAB, которая вычисляется по формуле $2 * S / d$, где S – площадь треугольника OAB.

Пересечение отрезков на прямой. Пусть $[x_1, x_2]$ и $[x_3, x_4]$ – два отрезка на прямой ($x_1 < x_2$, $x_3 < x_4$). Отрезки не пересекаются, если имеет место одно из двух расположений:

Если $x_3 > x_2$, то должно быть $x_4 > x_1$. Если $x_1 > x_4$, то должно быть $x_2 > x_3$. Эти два условия можно объединить в одно. Отрезки не пересекаются тогда и только тогда, когда имеет место неравенство: $(x_3 - x_2) * (x_4 - x_1) > 0$. Если это произведение неположительно, то отрезки пересекаются.

Мера пересечения отрезков на прямой. Пусть $[x_1, x_2]$ и $[x_3, x_4]$ – два пересекающихся отрезка.

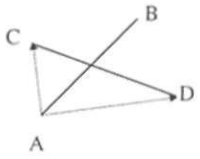
Пусть *left* = max(x_1, x_3) – максимум левых концов, *right* = min(x_2, x_4) – минимум правых. Тогда значение *right* – *left* будет длиной общей части отрезков.

Пересечение отрезков на плоскости. Пусть заданы два отрезка AB и CD координатами своих концов.

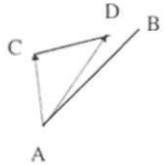
Ограничивающим прямоугольником геометрической фигуры называется наименьший из прямоугольников со сторонами, параллельными осям координат, который содержит в себе эту фигуру. Для отрезка с концами (x_1, y_1) и (x_2, y_2) ограничивающим будет прямоугольник с левым нижним углом (x_1', y_1') и правым верхним углом (x_2', y_2'), где $x_1' = \min(x_1, x_2)$, $y_1' = \min(y_1, y_2)$, $x_2' = \max(x_1, x_2)$, $y_2' = \max(y_1, y_2)$.

Отрезки AB и CD пересекаются тогда и только тогда, когда:
1. Пересекаются прямоугольники, которые их ограничивают;
2. [AC × AB] * [AD × AB] ≤ 0;
3. [CA × CD] * [CB × CD] ≤ 0.

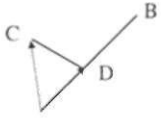
Ниже представлены различные случаи взаимного расположения двух отрезков и соответствующие значения векторных произведений:



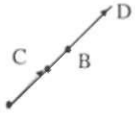
AC × AB < 0, AD × AB > 0



AC × AB < 0, AD × AB < 0

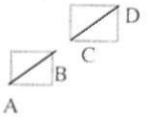


AC × AB < 0, AD × AB = 0



AC × AB = 0, AD × AB = 0

В следующем примере каждый из отрезков пересекает прямую, содержащую второй отрезок. Но при этом не пересекаются прямоугольники, их ограничивающие.



Пусть концы отрезков AB и CD имеют координаты: A(x_1,y_1), B(x_2,y_2), C(x_3,y_3), D(x_4,y_4). Функция RectanglesIntersects принимает в качестве аргументов 8 координат концов отрезков и выводит 1, если прямоугольники, ограничивающие отрезки AB и CD, пересекаются. Иначе функция возвращает 0.

```
int RectanglesIntersects(double x1,double y1,double x2,double y2,
                         double x3,double y3,double x4,double y4)
{
  if ((x3 - x2)*(x4 - x1) > 0) return 0;
  if ((y3 - y2)*(y4 - y1) > 0) return 0;
  return 1;
}
```


Функция intersect проверяет, пересекаются ли отрезки AB и CD. Сначала проверяется пересечение ограничивающих прямоугольников. Если они пересекаются, то строятся вектора AC, AB, AD, CA, CB, CD (например, координаты вектора CD содержатся в переменных CDx и CDy) и вычисляются векторные произведения, указанные в пунктах 2 и 3 условия пересечения отрезков. В зависимости от значений векторных произведений формируется возвращаемое значение. Оно равно 1, если отрезки AB и CD пересекаются и 0 иначе.

```
double intersect(double x1,double y1, double x2, double y2,
                 double x3,double y3, double x4, double y4)
{
  double ABx, ABy, ACx, ACy, ADx, ADy;
  double CAx, CAy, CBx, CBy, CDx, CDy;
  double ACxAB, ADxAB, CAxCD, CBxCD;
  if (!RectanglesIntersects(min(x1,x2),min(y1,y2),max(x1,x2),max(y1,y2),
      min(x3,x4),min(y3,y4),max(x3,x4),max(y3,y4))) return 0;
  ACx = x3 - x1; ACy = y3 - y1;
  ABx = x2 - x1; ABy = y2 - y1;
  ADx = x4 - x1; ADy = y4 - y1;

  CAx = x1 - x3; CAy = y1 - y3;
  CBx = x2 - x3; CBy = y2 - y3;
  CDx = x4 - x3; CDy = y4 - y3;

  ACxAB = ACx * ABy - ACy * ABx;
  ADxAB = ADx * ABy - ADy * ABx;
  CAxCD = CAx * CDy - CAy * CDx;
  CBxCD = CBx * CDy - CBy * CDx;

  return ACxAB * ADxAB <= 0 && CAxCD * CBxCD <= 0;
}
```

Точка пересечения отрезков. Пусть A(x_1, y_1), B(x_2, y_2) – концы отрезка AB, C(x_3, y_3), D(x_4, y_4) – концы отрезка CD. Запишем параметрические уравнения отрезков:

AB: $x(t_1) = x_1 + (x_2 - x_1) * t_1$, $y(t_1) = y_1 + (y_2 - y_1) * t_1$, $0 \le t_1 \le 1$,

CD: $x(t_2) = x_3 + (x_4 - x_3) * t_2$, $y(t_2) = y_3 + (y_4 - y_3) * t_2$, $0 \le t_2 \le 1$.

Отрезки пересекаются, если существуют такие t_1, t_2 ($0 \le t_1, t_2 \le 1$), что

$$x(t_1) = x(t_2),\ y(t_1) = y(t_2)$$

Или то же самое, что

$$x_1 + (x_2 - x_1) * t_1 = x_3 + (x_4 - x_3) * t_2$$
$$y_1 + (y_2 - y_1) * t_1 = y_3 + (y_4 - y_3) * t_2$$

Имеем систему линейных уравнений относительно t_1 и t_2:

$$\begin{cases} (x_2 - x_1)t_1 + (x_3 - x_4)t_2 = x_3 - x_1 \\ (y_2 - y_1)t_1 + (y_3 - y_4)t_2 = y_3 - y_1 \end{cases},$$

которую решаем методом Крамера.

Серединный перпендикуляр. Пусть A(x_1, y_1), B(x_2, y_2) – координаты концов отрезка, $ax + by + c = 0$ уравнение серединного перпендикуляра к нему. Поскольку вектора AB($x_2 - x_1$, $y_2 - y_1$) и (a, b) колинеарны, то

$$a = x_2 - x_1,\ b = y_2 - y_1$$

Серединный перпендикуляр проходит через точку $M\left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}\right)$ – середину отрезка AB, значит $ax + by + c = (x_2 - x_1)\frac{x_1 + x_2}{2} + (y_2 - y_1)\frac{y_1 + y_2}{2} + c = 0$, откуда $c = \frac{x_1^2 - x_2^2 + y_1^2 - y_2^2}{2}$.

```
void midperpend(double x1, double y1, double x2, double y2,
                double *a, double *b, double *c)
{
  *a = x2 - x1;
  *b = y2 - y1;
  *c = (x1*x1 - x2*x2 + y1*y1 - y2*y2) / 2;
}
```

Уравнение окружности с центром в точке (a, b) и радиусом r имеет вид:

$$(x-a)^2 + (y-b)^2 = r^2$$

Полярная система координат. Каждая точка в полярной системе координат определяется радиальной координатой r и полярным углом φ. Координата r соответствует расстоянию до полюса, а координата φ равна углу в направлении против часовой стрелки от полярной оси (луча, проходящего через 0°).

Связь полярных координат (r, φ) с Декартовыми (x, y): $\begin{cases} x = r\cos\varphi \\ y = r\sin\varphi \end{cases}$, $r^2 = x^2 + y^2$.

Полярный угол. Пусть A(x, y) – точка в декартовой системе координат. Для вычисления ее полярного угла φ можно воспользоваться соотношениями:

$$\varphi = \begin{cases} arctg\dfrac{y}{x}, x > 0, y \ge 0 \\ arctg\dfrac{y}{x} + 2\pi, x > 0, y < 0 \\ arctg\dfrac{y}{x} + \pi, x < 0 \\ \dfrac{\pi}{2}, x = 0, y > 0 \\ \dfrac{3\pi}{2}, x = 0, y < 0 \end{cases} \qquad \varphi = \begin{cases} arctg\dfrac{y}{x}, x > 0 \\ arctg\dfrac{y}{x} + \pi, x < 0, y \ge 0 \\ arctg\dfrac{y}{x} - \pi, x < 0, y < 0 \\ \dfrac{\pi}{2}, x = 0, y > 0 \\ -\dfrac{\pi}{2}, x = 0, y < 0 \end{cases}$$

$$\varphi \in [0; 2\pi) \qquad \varphi \in (-\pi; \pi]$$

Вычисление полярного угла. Функция PolarAngle вычисляет величину угла вектора p с осью абсцисс. Значение угла изменяется в пределах от 0 включительно до 2π не включительно.

```
double PolarAngle(pair<int,int> p)
// p != (0,0)
{
  double res = 0;
  int x = p.first, y = p.second;
  if (x == 0)
  {
    if (y > 0)res = PI / 2; else res = 3*PI/2;
  }
  else
  {
    res = atan(1.0*y/x);
    if (x < 0) res = res + PI;
    if (res < 0) res = res + 2*PI;
  }
  return res;
}
```

Функция atan(x) вычисляет арктангенс x в пределах от $-\pi/2$ до $\pi/2$.

Функция atan2(double y, double x) вычисляет значение арктангенса y/x в пределах от $-\pi$ до π. Если $x = 0$, или оба параметра равны нулю, то функция возвращает 0. С использованием atan2 функцию PolarAngle можно переписать в виде:

```
double PolarAngle(pair<int,int> p)
{
  double res = atan2(1.0*p.second,1.0*p.first);
  if (res < 0) res += 2*PI;
  return res;
}
```

Пусть A(x_1, y_1) и B(x_2, y_2) – две точки на плоскости, O – начало координат. Вектор OA образует больший угол с осью абсцисс, нежели OB, если OA × OB < 0, и при этом оба угла принадлежат либо интервалу $[0; \pi)$ – верхняя полуплоскость, либо $[\pi; 2\pi)$ – нижняя полуплоскость. Рассмотрим функцию сортировки точек по полярному углу от $-\pi$ до π.

```
int f(pair<int,int> x, pair<int,int> y)
{
  // xx = 1 если полярный угол x принадлежит [0; PI)
  int xx = (x.second > 0) || ((x.second == 0) && (x.first > 0));
  // yy = 1 если полярный угол y принадлежит [0; PI)
  int yy = (y.second > 0) || ((y.second == 0) && (y.first > 0));
  // если полярные углы лежат в разных полуплоскостях (верхней, нижней),
  // то меньшим считается угол, лежащий в нижней полуплоскости
  if (xx != yy) return xx < yy;
  // точки сортируются по полярному углу в промежутке [-PI; PI)
  return x.first * y.second - x.second * y.first > 0;
}
```

Следующая функция сортирует точки по полярному углу от 0 до 2π с использованием функции PolarAngle. Если точки имеют одинаковый полярный угол, то они сортируются по возрастанию расстояния от центра координат.

```
int len2(pair<int,int> p)
{
  return p.first * p.first + p.second * p.second;
}

int f(pair<int,int> x, pair<int,int> y)
{
  double xx = PolarAngle(x);
  double yy = PolarAngle(y);
  if (fabs(xx - yy) < 1e-7) return len2(x) < len2(y);
  return xx < yy;
}
```

Площадь треугольника ABC, заданного координатами вершин A(x_1, y_1), B(x_2, y_2), C(x_3, y_3), равна

$$S = \frac{1}{2} abs \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

Вычтем со второй и из третьей строки первую и распишем определитель по третьему столбцу:

$$S = \frac{1}{2} abs \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 - x_1 & y_2 - y_1 & 0 \\ x_3 - x_1 & y_3 - y_1 & 0 \end{vmatrix} = \frac{1}{2} abs \begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix} =$$

$$\frac{1}{2}\left|(x_2 - x_1) * (y_3 - y_1) - (x_3 - x_1) * (y_2 - y_1)\right|$$

```
double TriangleArea(int x1, int y1, int x2, int y2, int x3, int y3)
{
  return abs((x2 - x1) * (y3 - y1) - (x3 - x1) * (y2 - y1)) / 2.0;
}
```

Принадлежность точки треугольнику. Точка O лежит внутри треугольника ABC тогда и только тогда, когда все три поворота OAB, OBC и OCA правые (треугольник ABC ориентирован по часовой стрелке) или левые (треугольник ABC ориентирован против часовой стрелки). То есть все выражения OA × AB, OB × BC, OC × CA имеют одинаковый знак.

Относительное расположение треугольников. Треугольники ABC и DEF лежат по разные стороны от прямой AB тогда и только тогда, когда одновременно выполняются следующие три условия:

1) вершины C и D лежат по разные стороны от прямой AB: (AB × AC) * (AB × AD) < 0
2) вершины C и E лежат по разные стороны от прямой AB: (AB × AC) * (AB × AE) < 0
3) вершины C и F лежат по разные стороны от прямой AB: (AB × AC) * (AB × AF) < 0

Пересечение треугольников. Два треугольника не пересекаются тогда и только тогда, когда существует такая прямая, проходящая через одну из шести сторон треугольников, что плоскости треугольников лежат по разные стороны от нее.

Площадь выпуклого многоугольника. Пусть (x_1, y_1), (x_2, y_2), …, (x_n, y_n) – координаты вершин выпуклого многоугольника, заданные в порядке его обхода по или против часовой стрелки. Тогда площадь многоугольника вычисляется по формуле трапеций:

$$S = \text{abs}\left(\sum_{i=1}^{n} \frac{y_{i+1} + y_i}{2}(x_{i+1} - x_i)\right)$$

где $(x_{n+1}, y_{n+1}) = (x_1, y_1)$. Причем значение суммы положительно, если координаты точек заданы в порядке обхода по часовой стрелке, и отрицательно если против.

Пусть массив x содержит абсцисы точек, а y – ординаты, заданные в порядке обхода по или против часовой стрелки. Функция findArea вычисляет площадь многоугольника.

```
double findArea(vector<int> x, vector<int> y)
{
  double s;
  int i;
  x.push_back(x[0]); y.push_back(y[0]);
  for(s = i = 0; i < x.size() - 1; i++)
    s += (y[i+1] + y[i]) * (x[i+1] - x[i]) / 2.0;
  return fabs(s);
}
```

Уравнение окружности через три точки. Пусть $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$ – три точки, не лежащие на одной прямой. Тогда уравнение окружности, проходящей через три точки, имеет вид:

$$\begin{vmatrix} x^2 + y^2 & x & y & 1 \\ x_1^2 + y_1^2 & x_1 & y_1 & 1 \\ x_2^2 + y_2^2 & x_2 & y_2 & 1 \\ x_3^2 + y_3^2 & x_3 & y_3 & 1 \end{vmatrix} = 0$$

Уравнение можно также представить в виде $a * (x^2 + y^2) - bx + cy - d = 0$, где

$$a = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}, b = \begin{vmatrix} x_1^2 + y_1^2 & y_1 & 1 \\ x_2^2 + y_2^2 & y_2 & 1 \\ x_3^2 + y_3^2 & y_3 & 1 \end{vmatrix}, c = \begin{vmatrix} x_1^2 + y_1^2 & x_1 & 1 \\ x_2^2 + y_2^2 & x_2 & 1 \\ x_3^2 + y_3^2 & x_3 & 1 \end{vmatrix}, d = \begin{vmatrix} x_1^2 + y_1^2 & x_1 & y_1 \\ x_2^2 + y_2^2 & x_2 & y_2 \\ x_3^2 + y_3^2 & x_3 & y_3 \end{vmatrix}$$

Центр (x_c, y_c) и радиус r окружности, описанной вокруг треугольника ABC, находится из соотношений:

$$x_c = b / 2a,\ y_c = -c / 2a,\ r^2 = (b^2 + c^2 - 4ad) / 4a^2$$

Другой метод решения задачи состоит в написании уравнений двух серединных перпендикуляров к отрезкам AB и AC. Точка их пересечения, которую находим методом Крамера, и является центром искомой окружности.

Центр тяжести. Понятие "центр тяжести многоугольника" можно интерпретировать тремя различными способами:

1. Масса находится только в вершинах.
2. Масса равномерно распределена по границе многоугольника.
3. Масса равномерно распределена по области, ограниченной многоугольником.

Рассмотрим каждый из этих случаем в отдельности.

Масса находится только в вершинах. Пусть многоугольник состоит из n вершин, (x_i, y_i) – координаты i -ой вершины, m_i – ее масса. Пусть M – масса всех вершин. Тогда координаты центра тяжести определяются по формулам:

$$X_c = (m_1 * x_1 + \ldots + m_n * x_n) / M = \frac{1}{M}\sum_{i=1}^{n} m_i \cdot x_i$$

$$Y_c = (m_1 * y_1 + \ldots + m_n * y_n) / M = \frac{1}{M}\sum_{i=1}^{n} m_i \cdot y_i$$

Если каждая вершина весит одинаково, то формулы примут следующий вид:

$$X_c = (x_1 + \ldots + x_n) / n = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$Y_c = (y_1 + \ldots + y_n) / n = \frac{1}{n}\sum_{i=1}^{n} y_i$$

Масса равномерно распределена по границе многоугольника. В этом случае масса ребра пропорциональна его длине. Каждое ребро мы можем заменить на точечную массу, пропорциональную длине ребра. Обозначим через l_i длину i - го ребра. Пусть P – периметр многоугольника, $P = l_1 + \ldots + l_n$. Тогда формулы координат центра тяжести имеют вид:

$$X_c = (l_1 * x_1 + \ldots + l_n * x_n) / P = \frac{1}{P}\sum_{i=1}^{n} l_i \cdot x_i$$

$$Y_c = (l_1 * y_1 + \ldots + l_n * y_n) / P = \frac{1}{P}\sum_{i=1}^{n} l_i \cdot y_i$$

Здесь через (x_i, y_i) обозначены координаты середины i - го ребра.

Масса равномерно распределена по области, ограниченной многоугольником.

Теорема. Пусть фигура Ф является объединением двух других фигур Φ_1 и Φ_2, пересекающимися только по границе. Тогда центр тяжести фигуры Ф выражается так:

$$X_c = \frac{X_{c1} \cdot S_1 + X_{c2} \cdot S_2}{S}, \; Y_c = \frac{Y_{c1} \cdot S_1 + Y_{c2} \cdot S_2}{S}, \text{ где}$$

(X_c, Y_c) – координаты центра тяжести фигуры Ф;
(X_{c1}, Y_{c1}) – координаты центра тяжести фигуры Φ_1;
(X_{c2}, Y_{c2}) – координаты центра тяжести фигуры Φ_2;
S – площадь фигуры Ф, S_1 – площадь фигуры Φ_1, S_2 – площадь фигуры Φ_2.

Кроме того, для треугольника координаты центра тяжести определяются так:

$$X_c = \frac{x_1 + x_2 + x_3}{3}, \; Y_c = \frac{y_1 + y_2 + y_3}{3}.$$

Разобъем многоугольник на треугольники. Для каждого треугольника найдем его центр тяжести (x_{ci}, y_{ci}) и площадь s_i. Тогда координаты центра тяжести многоугольника можно найти следующим образом:

$$X_c = \frac{X_{c1} \cdot S_1 + \ldots + X_{cm} \cdot S_m}{S}, \; Y_c = \frac{Y_{c1} \cdot S_1 + \ldots + Y_{cm} \cdot S_m}{S}$$

Здесь m равно количеству треугольников, на которые разбит многоугольник.

Если вершин многоугольника заданы координатами (x_0, y_0), (x_1, y_1), …, (x_{n-1}, y_{n-1}), то координаты его центра тяжести вычисляются по формулам

$$X_c = \frac{1}{6S} \sum_{i=0}^{n-1} (x_i + x_{i+1}) \cdot (x_i y_{i+1} - x_{i+1} y_i),$$

$$Y_c = \frac{1}{6S} \sum_{i=0}^{n-1} (y_i + y_{i+1}) \cdot (x_i y_{i+1} - x_{i+1} y_i)$$

Здесь $(x_n, y_n) = (x_0, y_0)$, а через

$$S = \frac{1}{2} \sum_{i=0}^{n-1} (x_i y_{i+1} - x_{i+1} y_i)$$

обозначена площадь многоугольника.

Константа π в языке C объявляется так:

```
const double pi = acos(-1.0);
```

Расстояние между точками на сфере. Рассмотрим сферу, на которой точки, задаются парами (*широта, долгота*) = (*latitude, longitude*), $-89 \le latitude \le 89$, $-179 \le longitude \le 179$. Кратчайшее расстояние от точки (lat_1, lon_1) до (lat_2, lon_2) на сфере радиуса *radius* равно

```
radius * acos(sin(lat1) * sin(lat2) + cos(lat1) * cos(lat2) * cos(lon1 - lon2))
```


Поворот на угол на плоскости. Выведем формулы поворота на плоскости на угол φ против часовой стрелки. Пусть z' и z – комплексные числа. Тогда $z' = e^{i\varphi} z$ или $x' + iy' = (\cos\varphi + i\sin\varphi)(x + iy) = x\cos\varphi - y\sin\varphi + i(x\sin\varphi + y\cos\varphi)$. Откуда получаем формулы поворота:

$$\begin{cases} x' = x\cos\varphi - y\sin\varphi \\ y' = x\sin\varphi + y\cos\varphi \end{cases}$$

```
#include <stdio.h>
#include <math.h>
#define PI acos(-1.0)
double x, y, fi, xx, yy;


int main(void)
{
  x = 10, y = 0, fi = 90; fi = PI/180 * fi;
  printf("%.4lf %.4lf\n",x,y);
  xx = x * cos(fi) - y*sin(fi);
  yy = x * sin(fi) + y*cos(fi);
  printf("%.4lf %.4lf\n",xx,yy);
  return 0;
}
```

Пример. Формула поворота на 90 градусов против часовой стрелки имеют вид:

$$\begin{cases} x' = x\cos 90^\circ - y\sin 90^\circ \\ y' = x\sin 90^\circ + y\cos 90^\circ \end{cases} \text{ или } \begin{cases} x' = -y \\ y' = x \end{cases}$$

Отрезок на решетке. Рассмотрим отрезок, концы которого имеют целочисленные координаты (x_1, y_1) – (x_2, y_2). Тогда количество целочисленных точек, лежащих на отрезке, равно

$$1 + НОД(|x_1 - x_2|, |y_1 - y_2|)$$

Пример. Отрезок (1, 0) – (5, 2) содержит $1 + НОД(|5 - 1|, |2 - 0|) = 1 + НОД(4, 2) = 3$ целочисленные точки.

Теорема Пика. Площадь S многоугольника с целочисленными вершинами равна сумме

$$B + G / 2 - 1,$$

где B равно количеству целочисленных точек внутри многоугольника, а G количеству целочисленных точек на границе многоугольника

4. Две окружности

Определить количество точек пересечения двух окружностей.

Вход. 6 чисел x_1, y_1, r_1, x_2, y_2, r_2, где x_1, y_1, r_1, x_2, y_2 – координаты центров окружностей, а r_1, r_2 – их радиусы. Все числа – действительные, не превышают по модулю 1000000000, заданы не более чем с 3-мя знаками после запятой.

Выход. Количество точек пересечения. Если точек пересечения бесконечно много, то вывести -1.

Пример входа
```
0 0 5 5 0 5
```

Пример выхода
```
2
```

2129. Полярный угол точки

Найдите полярный угол точки.

Вход. Два целых числа – декартовы координаты точки, не совпадающей с началом координат. Входные числа не превышают по модулю 10000.

Выход. Одно действительное число – величина полярного угла входной точки в радианах, которая находится в интервале $[0; 2\pi)$. Ответ округлить с точностью до 6 знаков после запятой.

Пример входа
```
2 3
```

Пример выхода
```
0.982794
```

2130. Угол между векторами

Вычислить угол между двумя векторами.

Вход. Четыре целых числа – координаты двух ненулевых векторов. Все входные числа не превышают по модулю 10000.

Выход. Одно число – величина неориентированного угла между векторами с точностью до пяти десятичных знаков. Выводимое число должно принадлежать интервалу $[0; \pi]$.

Пример входа
```
2 1 3 5
```

Пример выхода
```
0.56673
```


2131. Длина вектора

Вычислить длину вектора.

Вход. Четыре целых числа x_1, y_1, x_2, y_2 – координаты начала и конца вектора соответственно. Все входные числа не превышают по модулю 10000.

Выход. Одно число – длина заданного вектора с точностью до шести десятичных знаков.

Пример входа
```
1 1 2 2
```

Пример выхода
```
1.414214
```

2132. Принадлежность точки прямой

Определите, принадлежит ли точка прямой, заданной уравнением $Ax + By + C = 0$.

Вход. Пять чисел – координаты точки и коэффициенты A, B, C уравнения прямой (гарантируется, что A и B одновременно не равны 0).

Выход. Одно число – длина заданного вектора с точностью до шести десятичных знаков.

Пример входа
```
1 1 2 2
```

Пример выхода
```
1.414214
```

2133. Принадлежность точки лучу

Определите, принадлежит ли заданная точка лучу.

Вход. Шесть целых чисел – координаты точки и координаты начала и конца вектора. Все числа не превышают по модулю 10000.

Выход. Одна строка - "YES", если точка принадлежит лучу, и "NO" в противном случае.

Пример входа
```
1 6 3 7 5 8
```

Пример выхода
```
NO
```

СТАРШАЯ ЛИГА. УСЛОВИЯ ЗАДАЧ

839. Пересечение отрезков

Два отрезка на плоскости заданы целочисленными координатами своих концов в декартовой системе координат.

Требуется определить, существует ли у них общая точка.

Вход. В первой строке содержатся координаты первого конца первого отрезка, во второй – второго конца первого отрезка, в третьей и четвёртой - координаты концов второго отрезка. Координаты целые и по модулю не превосходят 10000.

Выход. Выводится слово "Yes", если общая точка есть, или слово "No" - в противном случае.

Пример входа

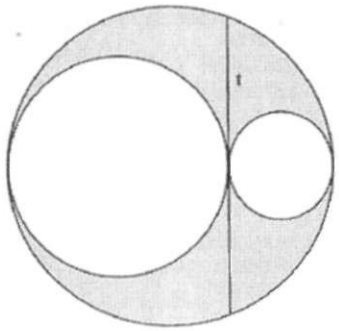
```
0 0
1 0
1 0
1 1
```

Пример выхода

```
Yes
```

1043. Геометрический парадокс

На рисунке ниже Вы видите два меньших круга, касающихся внешне друг друга. Больший круг расположен так, что он также касается этих двух кругов, но меньшие круги расположены внутри него. Хорда t большего круга является общей касательной к двум меньшим кругам радиусами r_1 и r_2. Центры всех трёх окружностей лежат на одной прямой. Вам заданны значения радиусов кругов r_1 и r_2, или значение длины хорды t. Вам необходимо вычислить значение площади большего круга, помеченной серым цветом на рисунке. Если заданной информации не достаточно для однозначного определения указанной площади, необходимо вывести сообщение "Impossible."



Вход. Первая строка содержит количество тестов n ($n \le 100$). Каждая следующая строка содержит или два целых числа (r_1 и r_2) или одно t. Все числа целые, меньше 100.

Выход. Для каждого набора входных данных выведите одну строку. Она должна содержать площадь серой области, если входных данных достаточно для её нахождения. В противном случае выведите единственное слово "Impossible." Площадь следует выводить с четырьмя десятичными знаками.

Пример входа

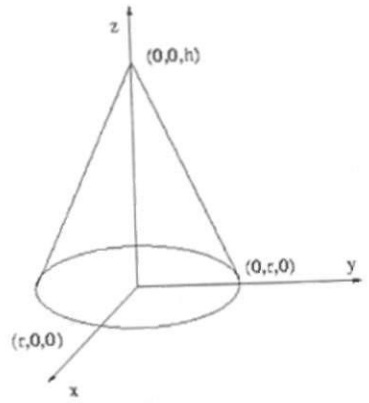
```
2
10 10
15 20
```

Пример выхода

```
628.3185
1884.9556
```

1501. Конусное расстояние

В трехмерном пространстве задан конус так, что его основание с радиусом r и центром в точке (0, 0, 0) лежит в плоскости $z = 0$. Вершина конуса расположена в точке (0, 0, h). На его поверхности заданы две точки в конусных координатах. Конусной координатой точки p называется пара чисел (d, A), где d – расстояние от вершины конуса до точки p, а A (A < 360) – угол в градусах между плоскостью y = 0 и плоскостью, проходящей через точки (0,0,0), (0, 0, h) и p, считая против часовой стрелки от направления оси x.



На поверхности конуса заданы две точки $p_1 = (d_1, A_1)$ и $p_2 = (d_2, A_2)$. Найти кратчайшее расстояние между p_1 и p_2, измеряемое по поверхности конуса.

Вход. Каждая строка содержит 6 действительных чисел: r, h, d_1, A_1, d_2, A_2.

Выход. Для каждого теста вывести наименьшее расстояние между точками p_1 и p_2 с двумя десятичными знаками.

Пример входа

```
3.0 4.0 2.0 0.0 4.0 0.0
3.0 4.0 2.0 90.0 4.0 0.0
6.0 8.0 2.14 75.2 9.58 114.3
3.0 4.0 5.0 0.0 5.0 90.0
```

Пример выхода

```
2.00
3.26
7.66
4.54
```

1502. Центр масс

Найти координаты центра масс выпуклого многоугольника, вырезанного из дерева.

Вход. Состоит из нескольких тестов. Первая строка каждого теста содержит количество вершин n ($n \leq 100$) многоугольника. Следующие n пар целых чисел описывают x и y координаты точек. Точки каждого многоугольника не упорядочены. Многоугольник в последнем тесте содержит $n < 3$ вершин и не обрабатывается.

Выход. Для каждого многоугольника в отдельной строке вывести координаты x и y центра масс.

Пример входа

```
4 0 1 1 1 0 0 1 0
3 1 2 1 0 0 0
7
-4 -4
-6 -3
-4 -10
-7 -12
-9 -8
-3 -6
-8 -3
1
```

Пример выхода

```
0.500 0.500
0.667 0.667
-6.102 -7.089
```

1503. Вписанные треугольники

На границе окружности с центром в начале координат и радиусом r заданы n различных точек. Поскольку все точки расположены на одной окружности, то любые три из них не колинеарны, и поэтому образуют треугольник. Вам необходимо вычислить суммарную площадь всех этих C_n^3 треугольников.

Вход. Состоит из не более чем 16 тестов. Каждый тест начинается начинается двумя целыми числами n ($0 \leq n \leq 500$) и r ($0 < r \leq 100$). Через n обозначено количество точек, а через r радиус окружности. Центр окружности находится в центре координат. Далее следуют n строк, каждая из которых содержит действительное число θ ($0.0 \leq \theta < 360.00$), которое определяет угол в градусах между точкой и направлением x-оси. Например, если θ равно 30.00 градусов, то соответствующая точка имеет декартовы координаты ($r \cdot \cos(30.00°)$, $r \cdot \sin(30.00°)$). Последняя строка содержит $n = r = 0$ и не обрабатывается.

Выход. Для каждого теста в отдельной строке вывести целое число – суммарную площадь (округленную до ближайшего целого) всех возможных треугольников, образованных заданными n точками.

Пример входа

```
5 10
10.00
100.00
300.00
310.00
320.00
3 20
10.00
100.00
300.00
0 0
```

Пример выхода

```
286
320
```

1504. Максимальный четырехугольник

В четырехугольнике, в который вписана окружность, известны длины двух соседних сторон и его периметр. Вычислить максимально возможное значение радиуса окружности.

Вход. Первая строка содержит количество тестов n ($n \leq 100$). Каждая следующая строка содержит три целых числа p, a, b, где a и b – длины соседних сторон, а p – периметр четырехугольника.

Выход. Для каждого теста вместе с его номером вывести в отдельной строке максимально возможное значение радиуса окружности с шестью точками после десятичной точки. Если такой окружности не существует, то вывести фразу Eta Shombhob Na., которая в переводе с языка Бангла обозначает «это не возможно».

Пример входа

```
2
20 5 6
20 10 12
```

Пример выхода

```
Case 1: 2.449490
Case 2: Eta Shombhob Na.
```

1505. Пересечение отрезков - 2

На декартовой плоскости задано n отрезков координатами своих концов. Определить, пересекаются ли они. Множество отрезков пересекается, если среди них существует хотя бы два, которые имеют как минимум одну общую точку.

Вход. Каждая строка содержит целочисленные координаты концов отрезка (x_1, y_1) - (x_2, y_2). Известно, что $n \le 6*10^5$ и $-1000 \le x_1, y_1, x_2, y_2 \le 1000$.

Выход. Вывести "intersect", если отрезки пересекаются и "NOT intersect" иначе.

Пример входа

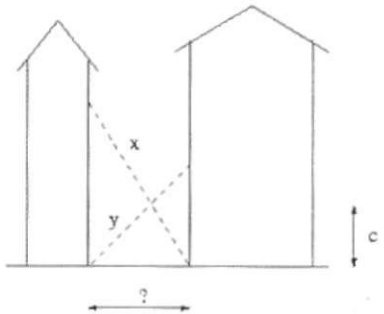
```
1 1 6 3
6 1 9 3
2 5 7 3
2 3 4 3
7 2 10 1
```

Пример выхода

```
Intersect
```

1506. Пересекающиеся лестницы

Вдоль узкой улицы по обе стороны расположены дома. Лестница длиной x футов расположена так, что ее низ упирается в основание дома на правой стороне, а верх прислонен к дому на левой стороне улицы. Аналогично низ лестницы длиной y футов упирается в основание дома на левой стороне, а верх прислонен к правому дому. Точка, в которой пересекаются лестницы, находится в точности c футов над землей. Какова ширина улицы?



Вход. Каждая строка содержит три положительных действительных числа x, y и c.

Выход. Для каждого теста вывести ширину улицы с тремя знаками после десятичной запятой.

Пример входа

```
30 40 10
12.619429 8.163332 3
10 10 3
10 10 1
```

Пример выхода

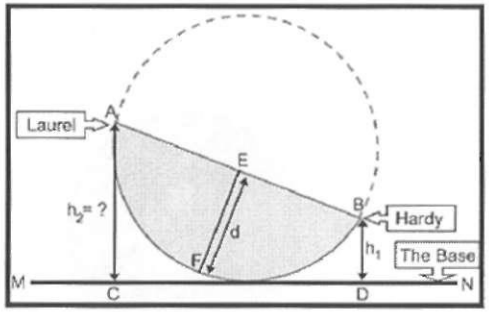
```
26.033
7.000
8.000
9.798
```


1507. История Лорела - Харди

Пила по дереву имеет вид сегмента AFB окружности радиуса r. Харди садится на конец B, а Лорел – на конец A (Харди тяжелее Лорела). Известно расстояние d = FE между серединой отрезка AB и серединой дуги AFB. Расстояние от точки B до земли равно h_1. Найти h_2 – расстояние от точки A до земли.

Вход. Первая строка содержит количество тестов n ($0 < n \le 1000$). Каждый тест содержит три числа r, d и h_1 ($10 \le r \le 100$, $5 \le d \le r$, $5 \le h_1 \le d$).

Выход. Для каждого теста вывести в отдельной строке его номер и величину h_2, округленную до четырех десятичных знаков.



Пример входа

```
2
10 10 10
10 7 6
```

Пример выхода

```
Case 1: 10.0000
Case 2: 8.0342
```

1508. Окна роз

Мистер Арнольд Геральд Ностик занимается разработкой главного окна нового городского собора. Окно круглое, его диаметр равен $2r$. Поскольку мистер А. Г. Ностик немножко знает о девственницах, святых и ангелах, он призадумался над геометрическим шаблоном: пусть n четное целое число, как минимум 4. Мистер Ностик планирует выбрать n точек, каждую на расстоянии r от центра окна, так чтобы они образовали правильный многоугольник (на картинке приведен пример с $n = 8$). Потом эти точки соединяются отрезками и полученные области закрашиваются как показано ниже (цвета выбираются произвольно). Заметим, что при $n = 8$ будет всего четыре области. Пронумеруем эти области 1, 2, 3 и 4 начиная с центральной. В общем случае образуется $n / 2$ областей.

Помогите мистеру Ностику узнать, сколько стекла каждого цвета необходимо приобрести для выкладки окна.

Вход. Первая строка содержит количество тестов t. Далее следуют t строк, каждая из которых содержит действительное число r ($1 \le r \le 100$), четное целое n ($4 \le n \le 40$), и k ($1 \le k \le n / 2$).

Выход. Для каждой входной тройки r, n, k в отдельной строке вывести площадь k-ой области окна, округленную до четырех десятичных знаков.

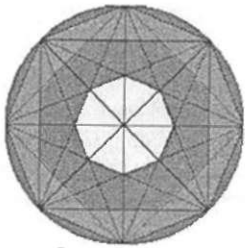
Пример входа

```
4
50 8 3
9.238794 8 2
10 4 1
20 4 1
```

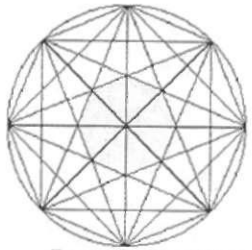
Пример выхода

```
2928.9322
100.0000
200.0000
800.0000
```

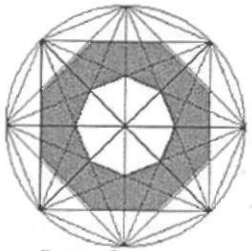
Правильный восьмиугольник в окружности



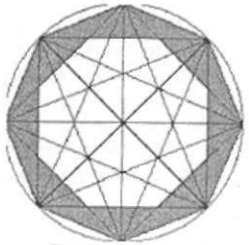
Окно роз с 8 точками



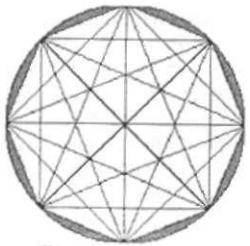
Первая область окна роз



Вторая область окна роз



Третья область окна роз



Четвертая область окна роз

1509. Раздел королевства

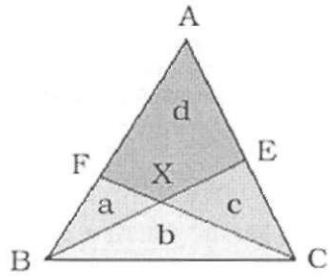
Король страны Геометрии в заботах. У него есть три сына, которые постоянно ссорятся. Король применял разные методы примерения, но все напрасно. И это его очень беспокоило.

"А что если разделить королевство?", – подумал король. Он пригласил советников и описал свой план. Король открыл карту.

Королевство имеет форму треугольника с вершинами 'A', 'B', 'C'. Король провел линию от B к E (E - произвольная точка на отрезке AC) и линию от C к F (F - произвольная точка на отрезке AB). Пересечение BE и CF обозначено через X.

Теперь образовалось четыре части - a (треугольник BFX), b (треугольник BCX), c (треугольник CEX) и d (четырехугольник AEXF). Король решил отдать области a, b, c трем сыновьям. А область d станет новым королевством.

Вы - главный советник. Король сообщает Вам значения a, b и c. Вам необходимо найти значение d. Если его найти невозможно, то сообщить об этом.



Вход. Состоит из не более чем 1000 тестов. Каждый тест содержит три неотрицательных действительных числа a, b, c (разделенных пробелом). Входные данные заканчиваются тестом у которого $a = -1$ и он не обрабатывается.

Выход. Для каждого теста вывести его номер, начиная с 1. В следующей строке вывести d (величина области королевства после раздела) округленное до 4 десятичных знаков или 'Poor King!' (без кавычек) если значение d определить невозможно. Формат выходных данных показан в примере.

Пример входа

```
1 2 1
2 4 2
1 3 3
-1 0 0
```

Пример выхода

```
Set 1:
2.0000
Set 2:
4.0000
Set 3:
5.0000
```

1510. Окружность через три точки

На плоскости заданы три точки, не лежащие на одной прямой. Необходимо найти уравнение окружности, проходящей через эти три точки. Уравнение окружности имеет один из следующих видов: $(x-a)^2+(y-b)^2=r^2$ или $x^2+y^2+cx+dy+e=0$.

Вход. Каждая строка содержит шесть действительных чисел A_x, A_y, B_x, B_y, C_x, C_y – координаты трех точек $A(A_x, A_y)$, $B(B_x, B_y)$, $C(C_x, C_y)$. Они являются действительными числами, разделенными одним или несколькими пробелами.

Выход. Для каждого теста необходимо вывести уравнение окружности в двух форматах как показано в примере. Значения *h*, *k*, *r*, *c*, *d* и *e* в уравнениях 1 и 2 необходимо выводить с тремя десятичными знаками. Если некоторое из значений *h*, *k*, *c*, *d*, *e* равно нулю, то соответствующее слагаемое выводить не следует. Знаки плюс и минус следует выводить по необходимости - так, чтобы избегать нескольких знаков перед числом. Знаки сложения, вычитания и равенства следует разделять от ближайших символов одним пробелом с каждой стороны. Никаких других лишних пробелов выводить не следует. После каждой пары уравнений следует выводить пустую строку. Формат вывода приведен в примере.

Пример входа

```
7.0 -5.0 -1.0 1.0 0.0 -6.0
1.0 7.0 8.0 6.0 7.0 -2.0
```

Пример выхода

```
(x - 3.000)^2 + (y + 2.000)^2 = 5.000^2
x^2 + y^2 - 6.000x + 4.000y - 12.000 = 0

(x - 3.921)^2 + (y - 2.447)^2 = 5.409^2
x^2 + y^2 - 7.842x - 4.895y - 7.895 = 0
```


МЛАДШАЯ ЛИГА. АНАЛИЗ ЗАДАЧ

4. Две окружности

Следует рассмотреть варианты внешнего и внутреннего касания, а также возможность расположения одной окружности внутри другой. Окружности имеют бесконечное количество точек пересечения только в случае их совпадения.

Пусть *dist2* – квадрат расстояния между центрами окружностей. Окружности касаются извне, если $(r_1+r_2)^2 = dist2$. Окружности имеют точку внутреннего касания, если $(r_1-r_2)^2 = dist2$. Если $(r_1+r_2)^2 < dist2$, то окружности не имеют общих точек и не содержатся друг в друге. Если $(r_1-r_2)^2 > dist2$, то окружности также не пересекаются, но при этом содержатся одна в другой.

2129. Полярный угол точки

Функция atan2(double *y*, double *x*) вычисляет значение арктангенса *y*/*x* в пределах от -π до π. Если $x = 0$, или оба параметра равны нулю, то функция возвращает 0. С помощью функции atan2 можно легко найти полярный угол точки.

2130. Угол между векторами

Пусть *a*(*ax*, *ay*) и *b*(*bx*, *by*) – два входных вектора. Вычислим модули этих векторов:

$$ma = |a| = \sqrt{ax^2+ay^2},\ mb = |b| = \sqrt{bx^2+by^2}$$

Вычислим скалярное произведение векторов: $(a, b) = sc = ax * bx + ay * by$.

Косинус угла между векторами равен $\frac{(a,b)}{|a|\cdot|b|}$, откуда находится и сам угол. Арккосинус угла лежит в интервале [0; π], поэтому никаких дополнительных вычислений производить не требуется.

2131. Длина вектора

Вектор *a* имеет координаты (x_2-x_1, y_2-y_1), его длина равна $|a| = \sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$.

2132. Принадлежность точки прямой

Точка (x_0, y_0) принадлежит прямой $Ax + By + C = 0$ тогда и только тогда, когда

$$Ax_0 + By_0 + C = 0$$

2133. Принадлежность точки лучу

Пусть $O(x_0, y_0)$ – координаты точки, $A(x_1, y_1)$ – координаты вершины луча, $B(x_2, y_2)$ – координаты точки, лежащей на луче. Сначала следует проверить, лежит ли точка O на прямой AB:

$$\frac{x_2 - x_1}{x_0 - x_1} = \frac{y_2 - y_1}{y_0 - y_1} \text{ или то же самое что } (x_0 - x_1) * (y_2 - y_1) = (x_2 - x_1) * (y_0 - y_1)$$

Точки O и B должны находиться по одну сторону от вершины луча A. Это условие следует записать как для координаты x, так и для y:

$$\frac{x_2 - x_1}{x_0 - x_1} \geq 0 \text{ и } \frac{y_2 - y_1}{y_0 - y_1} \geq 0$$

или для избежания деления то же самое что

$$(x_0 - x_1) * (x_2 - x_1) \geq 0 \text{ и } (y_0 - y_1) * (y_2 - y_1) \geq 0$$

СТАРШАЯ ЛИГА. АНАЛИЗ ЗАДАЧ

839. Пересечение отрезков

Отрезки AB и CD ***пересекаются*** тогда и только тогда, когда:

1. Пересекаются прямоугольники, которые их ограничивают;
2. $[AC \times AB] * [AD \times AB] \leq 0$;
3. $[CA \times CD] * [CB \times CD] \leq 0$.

Понятие ограничивающего прямоугольника и примеры взаимного расположения отрезков приведены в теоретическом материале.

1043. Геометрический парадокс

Решение задачи существует всегда, то есть фраза "Impossible." выводиться никогда не будет.

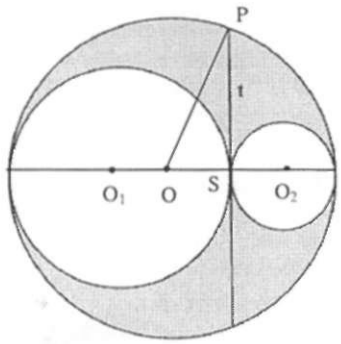
Радиус большей окружности равен $r = r_1 + r_2$. Если значения r_1 и r_2 известны, то площадь серой части равна $\pi((r_1 + r_2)^2 - r_1^2 - r_2^2) = 2\pi r_1 r_2$.

Пусть нам известно только значение t. Рассмотрим прямоугольный треугольник OPS. В нем $OP = r_1 + r_2$, $OS = r_1 + r_2 - 2r_2 = r_1 - r_2$, $SP = t / 2$. Запишем теорему Пифагора:

$$(r_1 + r_2)^2 = (r_1 - r_2)^2 + t^2 / 4$$

Раскрывая скобки, получим $2 r_1 r_2 = -2 r_1 r_2 + t^2 / 4$, $4 r_1 r_2 = t^2 / 4$.

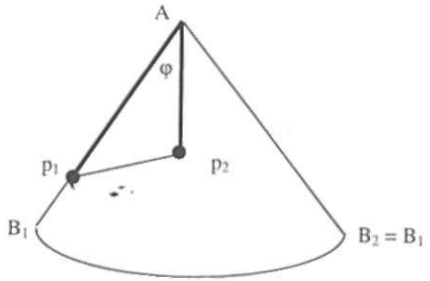
Площадь серой части равна $2\pi r_1 r_2 = \pi t^2 / 8$.



1501. Конусное расстояние

Переведем меру углов из градусной в радианную. Поскольку в 180 градусах содержится π радиан, то x градусам соответствует $x * \pi / 180$ радиан. Изобразим развертку конуса, разрезав его по прямой AB_1, которая проходит через вершину $A(0, 0, h)$ и точку p_1. Поскольку ищется минимальное расстояние между точками p_1 и p_2, то развертку отобразим той стороной, для которой $\angle p_1 A p_2 \leq \frac{1}{2}\angle B_1 A B_2$ (равенство достигается когда абсолютное значение разности $A_2 - A_1$ равно π). Обозначим $\varphi = \angle p_1 A p_2$. Поскольку длина образующей конуса l равна $AB_1 = \sqrt{r^2 + h^2}$, а длина дуги B_1B_2 развертки равна $2\pi r$, то $| B_1B_2 | = AB_1 * \angle B_1 A B_2$ или $2\pi r = l *$

$\angle B_1AB_2$. Здесь использована формула длины дуги $l = ra$, где l – длина дуги, r – радиус окружности, a – величина угла в радианах.



Обозначим через dA минимальное расстояние между углами A_1 и A_2. В развертке углу B_1AB_2 соответствует 2π радиан, а углу φ соответствует dA радиан. Имеем пропорцию:

$$\frac{\angle B_1AB_2}{\varphi} = \frac{2\pi}{dA}$$

Откуда $\varphi = \angle B_1AB_2 * dA / 2\pi = (2\pi r / l) * dA / 2\pi = r * dA / l$. Поскольку длины Ap_1 и Ap_2 известны, то из треугольника p_1Ap_2 по теореме косинусов находим длину отрезка p_1p_2. Она равна $\sqrt{d_1^2 + d_2^2 - 2d_1d_2\cos\varphi}$.

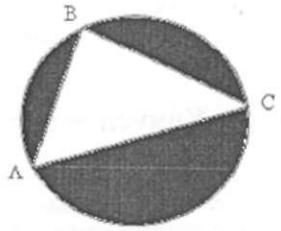
1502. Центр масс

Будем считать, что масса равномерно распределена по области, ограниченной многоугольником (то есть фигура вырезана из тонкого однородного материала).

Координаты центра тяжести многоугольника находятся по формулам, приведенным в теоретическом материале.

1503. Вписанные треугольники

Рассмотрим треугольник ABC. Вычислим его площадь как площадь круга минус площадь трех черных сегментов, как показано на рисунке.



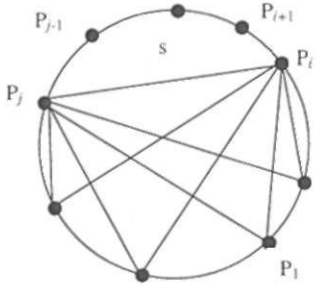
Таким образом, искомая площадь равна $C_n^3\pi r^2$ минус суммарная площадь сегментов, которую как далее будет показано, можно вычислить за время $O(n^2)$.

Напомним, что площадь сектора радиуса r, стягиваемого углом a, равна $\frac{1}{2}r^2a$. Площадь соответствующего сегмента равна площади сектора минус площадь равнобедренного треугольника с боковой стороной r и углом при вершине a. Таким образом, площадь сегмента равна

$$S_{сегмента} = \frac{1}{2}r^2a - \frac{1}{2}r^2\sin a = \frac{1}{2}r^2(a - \sin a)$$

Изначально положим искомую площадь всех треугольников равной $res = C_n^3\pi r^2 = \frac{n(n-1)(n-2)}{6}\pi r^2$. После чего из res будем последовательно вычитать площади сегментов.

Рассмотрим отрезок, соединяющий точки P_i и P_j ($i < j$). В переменной s вычислим площадь сегмента, который мы проходим при движении по окружности от точки P_i к P_j против часовой стрелки (по направлению возрастания углов).



Сегмент площади s вычитае P_n з общей площади res столько раз, сколько точек находится между P_i и P_j со стороны сегмента площади $\pi r^2 - s$. Сегмент площади s вычитается при подсчете площадей треугольников $P_iP_jP_k$, где k пробегает по точкам, расположенным от P_j до P_i при движении против часовой стрелки (k принимает значения $j + 1, j + 2, \ldots, i - 2, i - 1$). Таких точек будет $pnts1 = n - (j - i + 1)$. Вычитаем из res $pnts1$ раз площадь s.

Соответственно точек со стороны сегмента площади s будет $pnts2 = n - pnts1 - 2$. Именно столько раз необходимо вычитать площадь сегмента $\pi r^2 - s$ из res.

1504. Максимальный четырехугольник

Обозначим через c и d длины двух других сторон. Поскольку в четырехугольник вписана окружность, то $a + c = b + d$. Учитывая, что $a + b + c + d = p$, получим $a + c = b + d = p / 2$. Откуда $c = p / 2 - a$, $d = p / 2 - b$. Если длины сторон c и d будут не положительными, то искомого четырехугольника не существует.

Площадь четырехугольника выражается через периметр и радиус вписанной окружности по формуле: $s = (p/2) * r$. Радиус вписанной окружности будет максимальным, если будет максимальной площадь четырехугольника. Площадь четырехугольника выражается через длины его сторон a, b, c, d и противоположные углы B и D по формуле:

$$s = \sqrt{(p'-a)(p'-b)(p'-c)(p'-d) - abcd\cos^2\frac{B+D}{2}}$$, где $p' = p/2$ – полупериметр.

Распишем первый множитель: $p' - a = \frac{a+b+c+d}{2} - a = \frac{b+d+c-a}{2}$. Учитывая соотношение $a + c = b + d$, последнее выражение равно $\frac{a+c+c-a}{2} = c$. Аналогично $p' - b = \frac{a+b+c+d}{2} - b = \frac{a+c+d-b}{2} = \frac{b+d+d-b}{2} = d$, $p' - c = a$, $p' - d = b$. Формулу площади четырехугольника таким образом можно переписать в виде $s = \sqrt{abcd - abcd\cos^2\frac{B+D}{2}}$.

Площадь s будет максимальной, если $abcd\cos^2\frac{B+D}{2} = 0$, то есть при B + D = 180°. Сумма противоположных углов равна 180°, следовательно четырехугольник с заданными длинами сторон будет иметь максимальную площадь, если он вписан в окружность.

Искомая максимальная площадь равна $s = \sqrt{abcd}$.

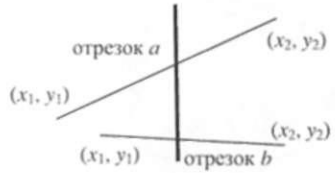
Пример. Для первого теста длины двух других сторон равны $c = 5$, $d = 4$. Они положительны, следовательно четырехугольник с заданными характеристиками существует и его площадь s равна $\sqrt{5*5*4*6} = 24.49490$. Радиус вписанной окружности равен $r = 2 * s / p = 2.449490$.

Для второго теста сумма двух сторон больше периметра, значит такого четырехугольника не существует.

1505. Пересечение отрезков - 2

Повернем все отрезки на малый угол чтобы ни один из них не был вертикальным. После считывания отрезка поменяем местами его концы если $x_2 < x_1$. Таким образом координаты обрабатываемых отрезков удовлетворяют соотношению $x_1 < x_2$. Создадим массив точек – концов отрезков, где с каждой точкой свяжем пару (номер отрезка которому принадлежит точка, тип конца отрезка – начало или конец). Отсортируем точки по возрастанию абсцисс.

Далее используем метод заметающей прямой. Двигаясь слева направо, будем обрабатывать концы отрезков. Отрезки, которые пересекает текущая заметающая прямая, будем хранить в мультимножестве s. Изначально оно пусто. Текущую абсциссу заметающей прямой храним в глобальной действительной переменной *xcur*. Реализуем функцию сравнения двух отрезков a и b, пересекающих прямую $x = xcur$. Для этого следует вычислить их y координаты пересечения с прямой $x = xcur$ и сравнить между собой. Теперь вставка отрезка в мультимножество s будет выполняться за логарифмическое время.



$x = xcur$

Пусть отрезок a пересекается с прямой $x = xcur$ в точке $(xcur, ya)$. Тогда имеет место пропорция:

$$\frac{xcur - a.x_1}{a.x_2 - a.x_1} = \frac{ya - a.y_1}{a.y_2 - a.y_1}$$

Откуда

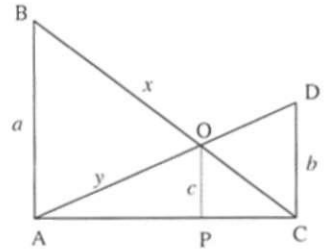
$$ya = a.y_1 + \frac{(xcur - a.x_1)(a.y_2 - a.y_1)}{a.x_2 - a.x_1}$$

Аналогично находится ордината точки пересечения отрезка b с прямой $x = xcur$.

События заметающей прямой обрабатываются следующим образом:

1. Если текущая точка – начало отрезка, то втавляем отрезок в мультимножество s и проверяем его на пересечение в верхним и нижним отрезком, находящимися в s.
2. Если текущая точка – конец отрезка, то удаляем отрезок из s, а отрезки, находившиеся выше и ниже его, проверяем на пересечение.

1506. Пересекающиеся лестницы



Треугольники AOP и ADC подобны: $\frac{c}{b} = \frac{AP}{AC}$.

Треугольники COP и CBA подобны: $\frac{c}{a} = \frac{CP}{CA}$.

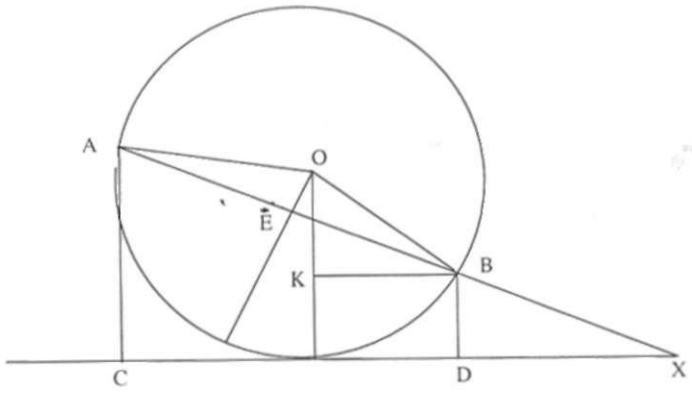
$\frac{c}{b} + \frac{c}{a} = \frac{AP}{AC} + \frac{CP}{AC} = \frac{AC}{AC} = 1$. Откуда $c\left(\frac{1}{a} + \frac{1}{b}\right) = 1$, $c = \frac{1}{\frac{1}{a} + \frac{1}{b}} = \frac{ab}{a+b}$.

Будем искать искомую ширину улицы d = AC бинарным поиском.

Изначально положим $0 \le d \le \min(x, y)$. По имеющимся d, x, y находим a, b и c. Чем больше d (при фиксированных x, y), тем меньше c.

1507. История Лорела – Харди

В треугольнике OEB: OB = r, OE = $r - d$, $\sin\angle EBO = (r - d) / r$. В треугольнике OKB: OB = r, OK = $r - h_1$, $\sin\angle KBO = (r - h_1) / r$. Находим $\angle KBA = \angle KBO - \angle EBO$. Из треугольника OEB: EB = $\sqrt{r^2 - (r-d)^2}$, AB = 2 * EB. Из треугольника BDX: BX = BD / $\sin\angle DXB = h_1 / \sin\angle KBA$. Находим AX = AB + BX. Из треугольника ACX находим ответ: h_2 = AC = AX * $\sin\angle DXB$.

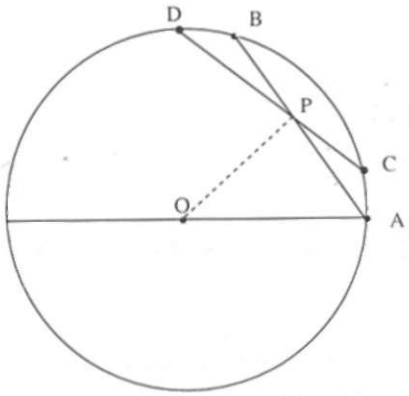


1508. Окна роз

k-ую область окна роз вместе со всеми внутренними областями будем называть k-ым многоугольником. Она в свою очередь будет представляет собой правильный n-угольник.

Радиусом окружности, описанной вокруг такого k-ого многоугольника, будет отрезок OP. Точка P образована пересечением отрезков AB и CD. При этом точки A и C, а также B и D идут последовательно по окружности, а между A и B лежит некоторое число точек. Упомянутые точки имеют координаты:

$$A(r,0),\ B\left(r\cos\frac{2\pi}{n}k, r\sin\frac{2\pi}{n}k\right),\ C\left(r\cos\frac{2\pi}{n}, r\sin\frac{2\pi}{n}\right),\ D\left(r\cos\frac{2\pi}{n}(k+1), r\sin\frac{2\pi}{n}(k+1)\right).$$



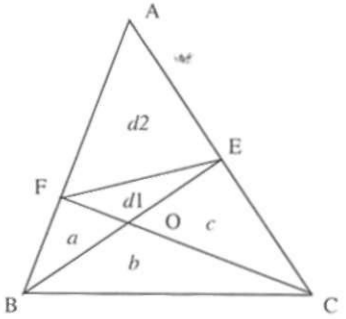
Строим уравнения прямых, проходящих через AB и CD. По правилу Крамера ищем точку их пересечения, то есть координаты точки P.

Площадь k-ого многоугольника равна n площадям равнобедренных треугольников с боковой стороной OP и углом при вершине $2\pi / n$. Их площади равны $\frac{1}{2}OP^2\sin\frac{2\pi}{n}$.

Искомую площадь k-ой области находим как разницу площадей k-ого многоугольника и $(k-1)$-го многоугольника.

1509. Раздел королевства

Искомую площадь d будем искать как сумму площадей $d1$ и $d2$.



Треугольники BFO и EFO имеют общее основание FO. Следовательно их площади $d1$ и a относятся как высоты, опущенные из вершин E и B на прямую FO. Аналогично треугольники BCO и ECO имеют общее основание CO. Значит их площади c и b относятся как высоты, опущенные из вершин E и B на прямую CO.

То есть $\frac{d1}{a}=\frac{c}{b}$, откуда $d1=\frac{a\cdot c}{b}$.

Теперь рассмотрим треугольники CAF и CBF с основаниями AF и BF. Они имеют одинаковую высоту, опущенную из вершины C на прямую AB. Следовательно площади этих треугольников относятся как длины сторон AF и BF.

Рассмотрим треугольники EAF и EBF с основаниями AF и BF. Они имеют одинаковую высоту, опущенную из вершины E на прямую AB. Площади этих треугольников относятся как длины сторон AF и BF. Имеем:

$$\frac{AF}{BF}=\frac{S_{CAF}}{S_{CBF}}=\frac{c+d1+d2}{a+b},$$

$$\frac{AF}{BF}=\frac{S_{EAF}}{S_{EBF}}=\frac{d2}{a+d1}$$

Следовательно $\frac{c+d1+d2}{a+b}=\frac{d2}{a+d1}$. Поскольку $d1$ уже найдено, то имеем равенство с одним неизвестным $d2$:

$$d2(a+b)=(c+d1+d2)(a+d1),\ d2(a+b)=(c+d1)(a+d1)+d2(a+d1),$$

$$d2(b-d1)=(c+d1)(a+d1),\ d2=\frac{(c+d1)(a+d1)}{b-d1}$$

Если $b\le d1$, то решения не существует.

1510. Окружность через три точки

Построим серединные перпендикуляры к отрезкам AB и AC. Точка их пересечения и будет центром искомой окружности O. Радиус окружности равен расстоянию OA.

МЛАДШАЯ ЛИГА. РЕАЛИЗАЦИЯ ЗАДАЧ

4. Две окружности

Изначально положим, что у нас имеются две точки пересечения.

```
int res = 2;
```

Читаем входные данные.

```
scanf("%lf %lf %lf %lf %lf %lf",&xx1,&yy1,&r1,&xx2,&yy2,&r2);
```

Вычисляем квадрат расстояния между центрами окружностей.

```
dist2 = (xx2-xx1)*(xx2-xx1) + (yy2-yy1)*(yy2-yy1);
```

Проверяем, не совпадают ли окружности.

```
if ((xx1 == xx2) && (yy1 == yy2) && (r1 == r2)) res = -1; else
```

Проверяем условие внешнего касания двух окружностей.

```
if (fabs((r1 + r2) * (r1 + r2) - dist2) < EPS) res = 1; else
```

Проверяем условие внутреннего касания двух окружностей.

```
if (fabs((r1 - r2) * (r1 - r2) - dist2) < EPS) res = 1; else
```

Рассматриваем случаи, когда окружности не имеют общих точек.

```
if ((r1 + r2)*(r1 + r2) < dist2 - EPS) res = 0; else
if ((r1 - r2)*(r1 - r2) > dist2 + EPS) res = 0;
```

Выводим ответ. Если ни одно из выше перечисленных условий не выполняется, то имеем две точки пересечения.

```
printf("%d\n",res);
```

2129. Полярный угол точки

Вычисляем полярный угол точки при помощи функции atan2. Если результатом выполнения функции atan2 будет отрицательное значение, то к нему следует прибавить 2π, так как ответ должен находиться в интервале $[0; 2\pi)$.

```
scanf("%lf %lf",&a,&b);
res = atan2(b,a);
if (res < 0) res += 2*PI;
printf("%.6lf\n",res);
```

2130. Угол между векторами

Вычисляем угол между двумя векторами по выше приведенной формуле.

```
scanf("%lf %lf %lf %lf",&ax,&ay,&bx,&by);
ma = sqrt(ax*ax + ay*ay); mb = sqrt(bx*bx + by*by);
sc = ax * bx + ay * by;
res = acos(sc / ma / mb);
printf("%.5lf\n",res);
```

2131. Длина вектора

Вычисляем длину вектора по выше приведенной формуле.

```
scanf("%lf %lf %lf %lf",&x_1,&y_1,&x_2,&y_2);
res = sqrt((x_2 - x_1)*(x_2 - x_1) + (y_2 - y_1)*(y_2 - y_1));
printf("%.6lf\n",res);
```

2132. Принадлежность точки прямой

Реализуем проверку принадлежности точки прямой, подставив ее координаты в уравнение прямой.

```
scanf("%d %d %d %d %d",&x,&y,&a,&b,&c);
if (a*x + b*y + c) printf("NO\n"); else printf("YES\n");
```

2133. Принадлежность точки лучу

Реализуем проверку принадлежности точки лучу согласно выше приведенным условиям.

```
scanf("%d %d %d %d %d %d",&x0,&y0,&x1,&y1,&x2,&y2);
if (((x0 - x1) * (y2 - y1) == (x2 - x1) * (y0 - y1)) &&
    ((x0 - x1) * (x2 - x1) >= 0)  && ((y0 - y1) * (y2 - y1) >= 0))
  printf("YES\n"); else printf("NO\n");
```


СТАРШАЯ ЛИГА. РЕАЛИЗАЦИЯ ЗАДАЧ

839. Пересечение отрезков

Поскольку входные координаты концов отрезка целые, то при вычислениях будем работать только с целочисленным типом long long. В дальнейшем нам необходима будет функция sgn, вычисляющая знак числа *n*.

```
int sgn(long long n)
{
  return (n > 0) - (n < 0);
}
```

Пусть концы отрезков AB и CD имеют координаты: A(x_1,y_1), B(x_2,y_2), C(x_3,y_3), D(x_4,y_4). Функция RectanglesIntersects принимает в качестве аргументов 8 координат концов отрезков и выводит 1, если прямоугольники, ограничивающие отрезки AB и CD, пересекаются. Иначе функция возвращает 0.

```
int RectanglesIntersects(long long x1,long long y1,long long x2,
      long long y2,long long x3,long long y3,long long x4,long long y4)
{
  if (sgn(x3 - x2) * sgn(x4 - x1) > 0) return 0;
  if (sgn(y3 - y2) * sgn(y4 - y1) > 0) return 0;
  return 1;
}
```

Функция intersect проверяет, пересекаются ли отрезки AB и CD. Ее описание приведено в теоретическом материале. Здесь мы приведем ее реализацию из-за того, что она работает с целочисленными координатами.

```
int intersect(long long x1,long long y1,long long x2,long long y2,
              long long x3,long long y3,long long x4,long long y4)
{
  long long ABx, ABy, ACx, ACy, ADx, ADy;
  long long CAx, CAy, CBx, CBy, CDx, CDy;
  long long ACxAB, ADxAB, CAxCD, CBxCD;
  if (!RectanglesIntersects(min(x1,x2),min(y1,y2),max(x1,x2),max(y1,y2),
      min(x3,x4),min(y3,y4),max(x3,x4),max(y3,y4))) return 0;

  ACx = x3 - x1; ACy = y3 - y1;
  ABx = x2 - x1; ABy = y2 - y1;
  ADx = x4 - x1; ADy = y4 - y1;

  CAx = x1 - x3; CAy = y1 - y3;
  CBx = x2 - x3; CBy = y2 - y3;
  CDx = x4 - x3; CDy = y4 - y3;

  ACxAB = ACx * ABy - ACy * ABx;
  ADxAB = ADx * ABy - ADy * ABx;
  CAxCD = CAx * CDy - CAy * CDx;
  CBxCD = CBx * CDy - CBy * CDx;
```

Во избежание переполнения следует сравнивать с нулем не произведение чисел, а произведение их знаков.

```
  if ((sgn(ACxAB) * sgn(ADxAB) > 0) || (sgn(CAxCD) * sgn(CBxCD) > 0))
    return 0;
  return 1;
}
```

Основная часть программы.

```
scanf("%lld %lld %lld %lld",&x1,&y1,&x2,&y2);
scanf("%lld %lld %lld %lld",&x3,&y3,&x4,&y4);
if (intersect(x1,y1,x2,y2,x3,y3,x4,y4)) printf("Yes\n");
else printf("No\n");
```

1043. Геометрический парадокс

Читаем входные данные.

```
scanf("%d",&n);
while(n--)
{
  scanf("%d",&r1);scanf("%c",&c); cnt = 1;
  if (c == ' ')
    scanf("%d\n",&r2), cnt = 2;
```

Вычисляем площадь серой области согласно выше приведенным формулам.

```
  if (cnt == 2) res = 2*PI*r1*r2; else res = PI * r1 * r1 / 8;
  printf("%.4lf\n",res);
}
```

1501. Конусное расстояние

Объявим константу π:

```
const double MY_PI = acos(-1.0);
```

Читаем входные данные, переводим меру углов из градусной в радианную. Вычисляем минимальное угловое расстояние *da* между p_1 и p_2.

```
while (scanf("%lf %lf %lf %lf %lf %lf",&r, &h, &d1, &a1, &d2, &a2) == 6)
{
  a1 *= MY_PI / 180; a2 *= MY_PI / 180;
  da = fabs(a2 - a1);
  if (da > MY_PI) da = 2*MY_PI - da;
```

Вычисляем длину образующей *l* и угол φ, на который отображается угол dA на развертке. По теореме косинусов находим расстояние между точками p_1 и p_2 и выводим результат.

```
  l = sqrt(r*r+h*h);
  fi = da * r / l;
  l = sqrt(d1*d1 + d2*d2 - 2*d1*d2*cos(fi));
  printf("%.2lf\n",l);
}
```

1502. Центр масс

Пусть пара *p* содержит координаты точки P. Пусть точка O – центр координат. Функция PolarAngle(*p*) возвращает значение полярного угла между вектором OP и осью OX, измеряемое в пределах от 0 до 2π радиан.


```
double PolarAngle(pair<double,double> p)
{
  double res = atan2(1.0*p.second,1.0*p.first);
  if (res < 0) res += 2*PI;
  return res;
}
```

Функция f используется при сортировке вершин многоугольника по полярному углу.

```
int f(pair<double,double> a, pair<double,double> b)
{
  return PolarAngle(a) < PolarAngle(b);
}
```

Функция TriangleSquare(*a*, *b*, *c*) вычисляет площадь треугольника, заданного координатами трех вершин.

```
double TriangleSquare(pair<double,double> a, pair<double,double> b,
                      pair<double,double> c)
{
  return abs((b.first - a.first) * (c.second - a.second) -
             (c.first - a.first) * (b.second - a.second)) / 2.0;
}
```

Основная часть программы. Координаты вершин очередного многоугольника запоминаем в векторе пар *v*.

```
while(scanf("%d",&n), n > 2)
{
  v.clear();
  for(i = 0; i < n; i++)
  {
    scanf("%d %d",&a,&b);
    v.push_back(make_pair(a,b));
  }
```

Найдем координаты точки (*NewXC*, *NewYC*), лежащей внутри многоугольника. Поскольку по условию задачи многоугольник выпуклый, то в качестве такой точки можно взять центр тяжести треугольника, образованного точками 0, 1 и 2.

```
  NewXC = (v[0].first + v[1].first + v[2].first) / 3.0;
  NewYC = (v[0].second + v[1].second + v[2].second) / 3.0;
```

Совершим параллельный перенос всех точек многоугольника на вектор (*–NewXC*, *–NewYC*).

```
  for(i = 0; i < n; i++)
    v[i].first -= NewXC, v[i].second -= NewYC;
```

Теперь начало координат (0, 0) находится внутри выпуклого многоугольника. Сортируем вершины многоугольника по полярному углу.

```
  sort(v.begin(),v.end(),f);
```

Разобъем исходный многоугольник на треугольники с вершинами (0, *i*, *i* + 1), $i = \overline{1, n-1}$. Для каждого такого тругольника вычисляем его площадь в переменной *tempSquare*, а также координаты его центра. Последние прибавляем к переменным *xc* и *yc*, умножая их предварительно на *tempSquare*. В переменной *s* накапливаем площадь всего многоугольника.

```
xc = yc = s = 0;
for(i = 1; i < n - 1; i++)
{
  tempSquare = TriangleSquare(v[0],v[i],v[i+1]);
  s += tempSquare;
  xc += tempSquare * (v[0].first + v[i].first + v[i+1].first) / 3.0;
  yc += tempSquare * (v[0].second + v[i].second + v[i+1].second) / 3.0;
}
```

Разделив полученные значения *xc* и *yc* на *s* и сдвинув их на вектор (*NewXC*, *NewYC*), получим координаты центра масс исходного многоугольника.

```
xc = xc / s + NewXC; yc = yc / s + NewYC;
if (fabs(xc) < 0.00001) xc = 0.0;
if (fabs(yc) < 0.00001) yc = 0.0;
```

Выводим координаты центра масс.

```
  printf("%.3lf %.3lf\n",xc,yc);
}
```

1503. Вписанные треугольники

Считаем градусную меру точек, переведем ее в радианную и занесем в ячейки массива d (d[*i*] содержит радианную меру *i* - ой точки).

```
double alfa, sq, d[MAX];
```

Пока не достигнем конца файла, читаем входные данные.

```
while(scanf("%d %d",&n,&r), n + r)
{
```

Заносим радианную меру точек в массив d, сортируем массив d по возрастанию.

```
  for(i = 0; i < n; i++) scanf("%lf",&d[i]), d[i] = d[i] * PI / 180;
  sort(d,d + n);
```

В переменную *res* изначально заносим площадь, равную площади C_n^3 кругов радиуса *r*, то есть значение $C_n^3\pi r^2 = \frac{n(n-1)(n-2)(n-3)}{6}\pi r^2$. Переменной *r2* присвоим значение $\frac{1}{2}r^2$, а *sq* площадь одного круга, то есть πr^2.

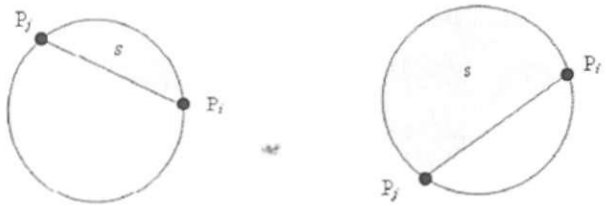
```
  res = n * (n - 1) * (n - 2) / 6 * PI * r * r;
  r2 = r * r / 2.0; sq = PI * r * r;
```

При помощи индексов *i* и *j* совершаем перебор пар точек.

```
  for(i = 0; i < n; i++)
    for(j = i + 1; j < n; j++)
    {
```

Вычисляем угол *alfa* между точками *i* и *j*, который равен d[*j*] – d[*i*]. Если *alfa* меньше π, то при движении от *i* к *j* мы проходим по меньшему сегменту и его площадь равна

$\frac{1}{2}r^2(alfa - \sin alfa)$. Иначе при движении от *i* к *j* мы проходим по большему сегменту и в таком случае его площадь равна $\pi r^2 - \frac{1}{2}r^2(alfa' - \sin alfa')$, где положено $alfa' = 2\pi - alfa$.



В обоих случаях переменной *s* присваивается площадь сегмента, который мы проходим при движении от P_i к P_j против часовой стрелки.

```
  alfa = d[j] - d[i];
  if (alfa < PI)
    s = r2 * (alfa - sin(alfa));
  else
  {
    alfa = 2 * PI - alfa,
    s = sq - r2 * (alfa - sin(alfa));
  }
```

Количество точек *points*, лежащих на сегменте, площадь которого $\pi r^2 - s$, равно $n - (j - i + 1)$. Таким образом из значения *res* необходимо вычесть *points* раз площадь сегмента *s*.

```
    points = n - (j - i + 1);
    res -= s * points;
```

Количество точек, лежащих на сегменте площади *s*, равно *points* = *n* – *points* – 2. Площадь противоположного сегмента равна *sq* – *s*. Остается вычесть ее из *res points* раз.

```
    points = n - points - 2;
    res -= (sq - s) * points;
  }
```

Выводим на печать искомую площадь *res*.

```
  printf("%.0lf\n",res);
}
```

1504. Максимальный четырехугольник

Для каждого теста читаем входные данные, вычисляем длины двух других сторон *c* и *d*.

```
scanf("%lf %lf %lf",&p,&a,&b);
c = p / 2 - a; d = p / 2 - b;
```

Если четырехугольник со сторонами *a*, *b*, *c*, *d* существует (их длины положительны), то вычисляем радиус вписанной окружности и выводим результат. Иначе выводим сообщение о невозможности построения четырехугольника.

```
if (c > 0 && d > 0)
{
  s = sqrt(a * b * c * d);
  r = 2 * s / p;
  printf("Case %d: %.6lf\n",i+1,r);
}
else printf("Case %d: Eta Shombhob Na.\n", i+1);
```

1505. Пересечение отрезков - 2

Объявим глобальную переменную *xcur*, которая будет содержать текущую абсциссу заметающей прямой. Объявим классы точки Point и отрезка Segment.

```
double xcur;
class Point
{
public:
  double x1, y1;
  Point(double x1 = 0, double y1 = 0)
  {
    this->x1 = x1; this->y1 = y1;
  }
};

class Segment
{
public:
  double x1, y1, x2, y2;
  Segment(double x1 = 0, double y1 = 0, double x2 = 0, double y2 = 0)
  {
    this->x1 = x1; this->y1 = y1;
    this->x2 = x2; this->y2 = y2;
  }
};
```

Функция сравнения двух отрезков *a* и *b*, пересекающих заметающую прямую $x = xcur$.

```
int operator< (Segment a, Segment b)
{
  double y1, y2;
  y1 = a.y1 + (xcur - a.x1) * (a.y2 - a.y1) / (a.x2 - a.x1);
  y2 = b.y1 + (xcur - b.x1) * (b.y2 - b.y1) / (b.x2 - b.x1);
  return y1 < y2;
}
```

Функция Intersect проверяет, пересекаются ли отрезки *a* и *b*.

```
int Intersect(Segment a, Segment b);
```

Мультимножество s содержит отрезки, пересекающие заметающую прямую $x = xcur$. Объявим также несколько итераторов.

```
multiset<Segment> s;
multiset<Segment>::iterator iter, iPrev, iNext;
```

Входные отрезки храним в векторе *v*. Множество концов отрезков с дополнительной информацией (номер отрезка которому принадлежит точка и тип точки – начало (0) или конец (1) отрезка) храним в векторе *p*.


```
vector<Segment> v;
vector<pair<Point,pair<int,int> > > p;
```

Функция сортировки точек по неубыванию абсцисс.

```
int f(pair<Point,pair<int,int> > a, pair<Point,pair<int,int> > b)
{
  if (a.first.x1 == b.first.x1) return a.second.second < b.second.second;
  return a.first.x1 < b.first.x1;
}
```

Основная часть программы. Читаем входные отрезки и совершаем их обработку перед запуском заметающей прямой.

```
n = 0;
while(scanf("%lf %lf %lf %lf",&x_1,&y_1,&x_2,&y_2) == 4)
{
```

Поворачиваем отрезок на угол EPS = 0.001 чтобы он не был вертикальным.

```
  x11 = x_1 * cos(EPS) - y_1*sin(EPS);
  y11 = x_1 * sin(EPS) + y_1*cos(EPS);
  x22 = x_2 * cos(EPS) - y_2*sin(EPS);
  y22 = x_2 * sin(EPS) + y_2*cos(EPS);
```

Первый конец каждого отрезка должен находиться не правее второго.

```
  if (x22 < x11) swap(x11,x22), swap(y11,y22);
  v.push_back(Segment(x11,y11,x22,y22));
  p.push_back(make_pair(Point(x11,y11),make_pair(n,0)));
  p.push_back(make_pair(Point(x22,y22),make_pair(n,1)));
  n++;
}
```

Сортируем концы отрезков по неубыванию абсцисс.

```
sort(p.begin(),p.end(),f);
```

Запускаем заметающую прямую. Она идет по абсциссам точек, хранящимся в массиве *p*.

```
flag = 0;
for(i = 0; i < p.size(); i++)
  {
```

Если текущая точка – начало отрезка, то вставляем этот отрезок в мультимножество *s* и проверяем, пересекается ли он с верхним и нижним отрезками в мультимножестве.

```
    if (p[i].second.second == 0)
    {
      xcur = p[i].first.x1 + 0.1;
      iter = s.insert(v[p[i].second.first]);
      iPrev = iNext = iter;
      if (iPrev != s.begin())
      {
        iPrev--;
        if (Intersect(*iter,*iPrev)) flag = 1;
      }
      iNext++;
```

```
      if (iNext != s.end())
      {
        if (Intersect((*iter),(*iNext))) flag = 1;
      }
    } else
    {
```

Если текущая точка – конец отрезка, то устанавливаем итератор *iter* на него. Проверям, пересекаются ли отрезки, находящиеся выше и ниже его. После чего удаляем из мультимножества *s* отрезок, конец которого обрабатываем.

```
      iter = s.find(v[p[i].second.first]);
      iPrev = iNext = iter; iNext++;
      if ((iPrev != s.begin()) && (iNext != s.end()))
      {
        iPrev--;
        if (Intersect((*iPrev),(*iNext))) flag = 1;
      }
      s.erase(iter);
    }
    if (flag) break;
  }
```

Выводим результат в зависимости от значения переменной *flag*.

```
  if (!flag) printf("NOT ");
  printf("intersect\n");
```

1506. Пересекающиеся лестницы

Читаем входные данные для каждого теста.

```
while(scanf("%lf %lf %lf",&x,&y,&c) == 3)
{
```

Установим *left* = 0, *right* = min(*x*,*y*). Далее в цикле будет иметь место неравенство $left \le d \le right$.

```
  left = 0;
  if (x < y) right = x; else right = y;
  d = (left + right) / 2;
  do
  {
```

Вычисляем значения *a*, *b*, *c*.

```
    a = sqrt(x*x - d*d); b = sqrt(y*y - d*d);
    c1 = 1/(1/a + 1/b);
```

Если найденное значение *c*1 меньше искомого *c*, то необходимо уменьшить верхнюю границу *d*. Иначе следует увеличить его нижнюю границу.

```
    if (c1 < c) right = (left + right) / 2;
          else left = (left + right) / 2;
    d = (left + right) / 2;
```


Вычисления проводим до требуемой в ответе точности (четыре десятичных знака).

```
  } while (fabs(c1 - c) > 0.00001);
```

Выводим результат.

```
  printf("%.3lf\n",d);
}
```

1507. История Лорела – Харди

Вычисляем $b = \angle KBA = \angle KBO - \angle EBO = \arcsin(r - h_1) / r - \arcsin(r - d) / r$. Если $b = 0$, то точки A и B находятся на одинаковом расстоянии от земли (прямой CX) и $h_2 = h_1$. Иначе вычисляем $AX = 2 * \sqrt{r^2 - (r-d)^2} + h_1 / \sin \angle KBA$ и находим ответ $h_2 = AX * \sin \angle DXB$ ($\angle DXB = \angle KBA$).

```
scanf("%d",&t);
for(i = 0; i < t; i++)
{
  scanf("%lf %lf %lf",&r,&d, &h1);
  printf("Case %d: ",i+1);
  b = asin((r - h1)/r) - asin((r-d)/r);
  if (b == 0.0)
  {
    printf("%0.4lf\n",h1);
    continue;
  }
  ax = 2 * sqrt1(r*r - (d-r)*(d-r)) + h1/sin(b);
  h2 = ax * sin(b);
  printf("%0.4lf\n",h2);
}
```

1508. Окна роз

Уравнением прямой, проходящей через точки (x_1, y_1) и (x_2, y_2), будет $ax + by + c = 0$.

```
void GetLine(double x1,double y1, double x2, double y2,
             double *a, double *b, double *c)
{
  *a = y2 - y1;
  *b = -(x2 - x1);
  *c = -x1*(y2 - y1) + y1*(x2 - x1);
}
```

Точкой пересечения прямых $a_1x + b_1y = c_1$ и $a_2x + b_2y = c_2$ будет P(*x*, *y*).

```
void GetPoint(double a1,double b1, double c1,
              double a2, double b2, double c2, double *x, double *y)
{
  double d = a1*b2 - a2*b1;
  *x = (c1*b2 - c2*b1) / d;
  *y = (a1*c2 - a2*c1) / d;
}
```

Вычисление площади k-ого многоугольника. Точка P имеет кординаты (x, y), а $d = OP^2$.

```
double Area(double x, double y)
{
  double d = x*x+y*y;
  return n*d*sin(2*PI/n)/2;
}

double FindSquare(int k)
{
  double a1,b1,c1,a2,b2,c2;
  double x,y;
  GetLine(r,0,r*cos(2*PI*k/n),r*sin(2*PI*k/n),&a1,&b1,&c1);
  GetLine(r*cos(2*PI/n),r*sin(2*PI/n),r*cos(2*PI*(k+1)/n),
                        r*sin(2*PI*(k+1)/n),&a2,&b2,&c2);
  GetPoint(a1,b1,-c1,a2,b2,-c2,&x,&y);
  return Area(x,y);
}
```

Основная часть программы.

```
PI = 2*acos(0.0);
scanf("%d",&m);
for(i = 0; i < m; i++)
{
  scanf("%lf %d %d",&r,&n,&k);
  if (k == n/2) UpperS = PI*r*r;
  else UpperS = FindSquare(n/2-k);
  DownS = FindSquare(n/2-k+1);
  Res = UpperS - DownS;
  printf("%.4lf\n",Res);
}
```

1509. Раздел королевства

Читаем входные данные для каждого теста, вычисляем значение площади $d1$.

```
while(scanf("%lf %lf %lf",&a,&b,&c), a != -1.0)
{
  d1 = a * c / b;
  printf("Set %d:\n",cs++);
```

Проверяем условие $b > d1$ и вычисляем площадь $d2$. Искомая площадь d равна $d1 + d2$.

```
  if (b > d1)
  {
    d2 = (a + d1) * (c + d1) / (b - d1);
    printf("%.4lf\n",d1+d2);
  }
  else printf("Poor King!\n",d1+d2);
}
```


1510. Окружность через три точки

Функция kramer решает систему линейных уравнений $\begin{cases} a_1 x + b_1 y = c_1 \\ a_2 x + b_2 y = c_2 \end{cases}$ методом Крамера

```
int kramer(double a1,double b1, double c1,
           double a2,double b2, double c2, double *x, double *y);
```

Функция midperpend по заданным координатам концов отрезка A(x_1, y_1) и B(x_2, y_2) строит серединный перпендикуляр $ax + by + c = 0$.

```
void midperpend(double x1, double y1, double x2, double y2,
                double *a, double *b, double *c);
```

Функция circle по трем точкам A(x_1, y_1), B(x_2, y_2) и C(x_3, y_3) находит центр окружности (xc, yc) и радиус r, который через них проходит. Для этого строятся серединные перпендикуляры к отрезкам AB и AC, после чего находится точка их пересечения O – центр искомой окружности. Радиус r вычисляется как расстояние между точками O и A.

```
void circle(double x1, double y1, double x2, double y2, double x3, double y3,
            double *xc, double *yc, double *r)
{
  double a1,b1,c1,a2,b2,c2;
  midperpend(x1,y1,x2,y2,&a1,&b1,&c1);
  midperpend(x1,y1,x3,y3,&a2,&b2,&c2);
  kramer(a1,b1,-c1,a2,b2,-c2,xc,yc);
  *r = sqrt((x1 - *xc)*(x1 - *xc) + (y1 - *yc)*(y1 - *yc));
}
```

Переменную епсилон EPS определим таким образом:

```
#define EPS 1e-6
```

Для каждого теста находим окружность, проходящую через три точки, и выводим данные о ней в соответствии с требуемым форматом.

```
while(scanf("%lf %lf %lf %lf %lf %lf",&x_1,&y_1,&x_2,&y_2,&x_3,&y_3) == 6)
{
  circle(x_1,y_1,x_2,y_2,x_3,y_3,&xc,&yc,&r);
  if (fabs(xc) < EPS) printf("x^2"); else
  {
    printf("(x");
    if (xc >= 0.0) printf(" - "); else printf(" + ");
    printf("%.3lf)^2",fabs(xc));
  }
  printf(" + ");
  if (fabs(yc) < EPS) printf("y^2"); else
  {
    printf("(y",fabs(xc));
    if (yc >= 0.0) printf(" - "); else printf(" + ");
    printf("%.3lf)^2",fabs(yc));
  }
  printf(" = %.3lf^2\n",r);
  printf("x^2 + y^2");
```

```
  if (fabs(xc) > EPS)
  {
    if (xc > 0.0) printf(" - "); else printf(" + ");
    printf("%.3lfx",2*fabs(xc));
   }
  if (fabs(yc) > EPS)
  {
    if (yc > 0.0) printf(" - "); else printf(" + ");
    printf("%.3lfy",2*fabs(yc));
  }
  r1 = xc*xc + yc*yc - r*r;
  if (fabs(r1) > EPS)
  {
    if (r1 > 0.0) printf(" + "); else printf(" - ");
    printf("%.3lf",fabs(r1));
  }
  printf(" = 0\n\n");
}
```