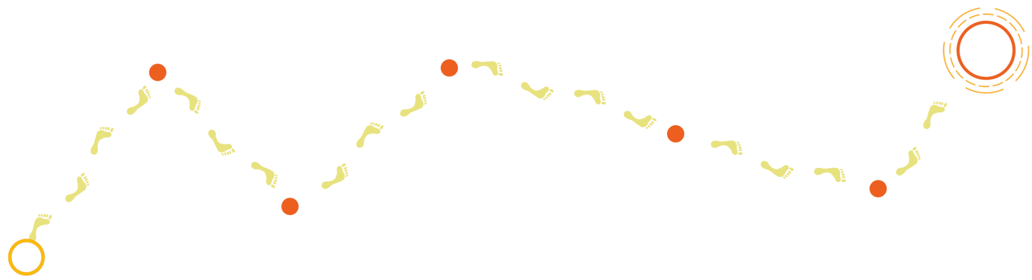
The Travelling Influencer Problem

Time Limit: 6 seconds



Payton is a social media influencer who likes to travel around the two-dimensional Cartesian plane to show all the nice places you can visit but probably cannot afford!

A *marketing* agency took interest in him and offered him a contract that stipulates that:

- He must visit exactly k (not necessarily distinct) locations. Label the ith location L_i (for each i from 1 to k).
- To keep his locations varied, for each $i = 3, \ldots, k$ location L_i must be:
  - exactly a_i units away from location L_{i-1}, and
  - exactly b_i units away from location L_{i-2}.

Moreover, he gets paid depending on how far L_1 is from L_k. This bonus is an amount of bitcoin equal to the distance of L_1 and L_k.

The agency lets Payton decide the value of k, the total number of locations he wants to visit and the coordinates of these locations. Hence, Payton can optimize the choice of his locations to make his bonus higher.

Let $n \geq 3$ be a positive integer. For every j from 3 to n, determine the maximum possible bitcoin Payton can receive as bonus if the contract stipulated that he must visit exactly $k = j$ locations. If there is no valid choice of locations (for a given k), we define the answer to be -1.

Your answer will be considered correct if it is within an absolute or relative error of 10^{-6} from the judge's answer.

Notes:

- **Important:** The test data are generated in a particular way. See the **Test Data** section below.
- We measure "distance" normally, that is, the (Euclidean) distance between two points (x_1, y_1) and (x_2, y_2) is

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

- The absolute error between a and b is $|a-b|$. The relative error is $\frac{|a-b|}{\max(|a|,|b|)}$ (which is only defined if a and b are not both zero).

Hints:

- You'll probably use floating points here. Be careful! There's a reason why floating-point problems don't commonly appear at the IOI.
- For example, to check whether three points are collinear, you should use a small tolerance $\varepsilon > 0$. (e.g., consider them collinear if the error from a true line is within a small error factor)
- Use `double` instead of `float` since it's more precise. For extra precision, use `long double` if you can (for `printf`/`scanf` users, you use `%Lf` instead of `%lf`), though beware that it's slower than `double`.

Input Format

The first line of input contains t, the number of test cases. The description of t test cases follow.

The first line of each test case contains an integer n. Then $n-2$ lines follow. The ith of these $n-2$ lines contains two space-separated integers a_{i+2} and b_{i+2}. In other words,

- line 1 contains a_3 and b_3,
- line 2 contains a_4 and b_4,
- line 3 contains a_5 and b_5,
- ...
- line $n-2$ contains a_n and b_n.

Output Format

For each test case, output $n-2$ lines, the ith of which contains a single real number denoting the answer for $k = i+2$. In other words,

- line 1 contains the answer for $k = 3$,
- line 2 contains the answer for $k = 4$,
- line 3 contains the answer for $k = 5$,
- ...
- line $n-2$ contains the answer for $k = n$.

Test Data

All test data, except the sample input, are generated in a particular way.

All t test cases in a file have the same n, and each test case is generated as follows:

- Choose 6 integers each randomly from the interval $[1234, 2345]$. (In particular, they may not be distinct.)
- For each i from 3 to n, choose each a_i and b_i randomly from these 6 integers. (In particular, integers that appear more than once among the 6 are more likely to be chosen.)

In addition, the number of test files for each subtask is specified: see the constraints.

Note: In the above, "randomly" means uniformly at random from among all possibilities, and independently of other choices.

Constraints and Subtasks

Here, let f be the number of test files in a subtask.

For all subtasks
$t \geq 1$ $1234 \leq a_i, b_i \leq 2345$ $n \geq 3$ $f \geq 1$

Subtask	Points	Constraints
1	**0**	$n = 7, t = 1, f = 1$ (This is the sample input)
2	**16**	$n = 7, t = 300, f = 4$
3	**6**	$n = 8, t = 300, f = 4$
4	**21**	$n = 18, t = 20, f = 8$
5	**31**	$n = 45, t = 10, f = 16$
6	**13**	$n = 270, t = 5, f = 32$
7	**13**	$n = 3000, t = 5, f = 32$

Important: For this problem, you can only get points in a subtask if you also solve all previous subtasks.

Sample I/O

Input	Output

`1`
`7`
`2222 1304`
`2291 2211`
`2222 1358`
`1358 1304`
`2222 1304`
	`1304.0000`
	`3595.0000`
	`2662.0000`
	`3500.4794`
	`3906.8231`

Explanation

In the sample input, we have $n = 7$. For $k = 7$, one optimal choice of locations is the following:

$$\begin{array}{ll} L_1 \approx (374.8582858, 746.9623715) & L_5 \approx (2571.929917, 2224.6479) \\ L_2 \approx (3477.573957, 2421.866946) & L_6 \approx (2391.457699, 878.6933025) \\ L_3 \approx (1527.560504, 1356.629162) & L_7 \approx (3828.388694, 2573.542357) \\ L_4 \approx (3511.624704, 211.1291623) & \end{array}$$

Here is an illustration of these 7 locations:

Thus, the distance between locations L_1 and L_7 is (approximately)

$$\sqrt{(374.8582858 - 3828.388694)^2 + (746.9623715 - 2573.542357)^2} \approx 3906.823099,$$

which can be shown to be the maximum possible among all choices. Hence, for $k = 7$, the maximum possible number of bitcoin Payton can receive as bonus is ≈ 3906.823099.