Problem 12: Alex was obliterated by a sonically-charged shriek
13 Point(s)

Problem ID: `warden2`
Rank: `4`

Introduction

This is the second of a two part problem series! You can find the first part here. Key differences introduced in this second part are highlighted for emphasis.

You are the Warden—terror of the deep dark and sentinel of the ancient city. After respawning, Steve brings along a friend this time: Alex! You must destroy both of them at all costs! Again, although unable to see the intruders, you can track them by pulsing vibrations through sculk sensors throughout the city. Expose their locations and obliterate them with a sonic boom!

Problem Statement

This is an interactive problem! Communicate with the judge using a series of *pulse* queries and *blast* queries. Using $\mathbf{P} = 1500$ or fewer *pulse*s, find both Steve and Alex and *blast* them to pass.

The Warden is at (0, 0) on the 2D coordinate plane. Steve is at $(\mathbf{X_S}, \mathbf{Y_S})$ and Alex is at $(\mathbf{X_A}, \mathbf{Y_A})$. Both of these are real number coordinates between $\mathbf{-10^5}$ **and** $\mathbf{10^5}$ predetermined for each test case, but not given to you as input. In an effort to not get blasted together, Steve and Alex are guaranteed to be at least a distance of 1000 apart on each axis.

To start, you can send a *pulse* to any real number coordinate (x_p, y_p) between $\mathbf{-10^6}$ **and** $\mathbf{10^6}$. Note that **this area is larger than the area where Steve and Alex may be**. When you *pulse*, the judge responds with the Euclidean distance of the following path as a decimal number:

$$\text{Warden} \Rightarrow \text{Pulse Location} \Rightarrow \text{Steve} \Rightarrow \text{Pulse Location} \Rightarrow \text{Alex} \Rightarrow \text{Pulse Location} \Rightarrow \text{Warden}$$

In other words, you will receive the value of:

$$d((0, 0), (x_p, y_p)) + d((x_p, y_p), (\mathbf{X_S}, \mathbf{Y_S})) + d((\mathbf{X_S}, \mathbf{Y_S}), (x_p, y_p)) + d((x_p, y_p), (\mathbf{X_A}, \mathbf{Y_A})) + d((\mathbf{X_A}, \mathbf{Y_A}), (x_p, y_p)) + d((x_p, y_p), (0, 0))$$

where d is the distance function:

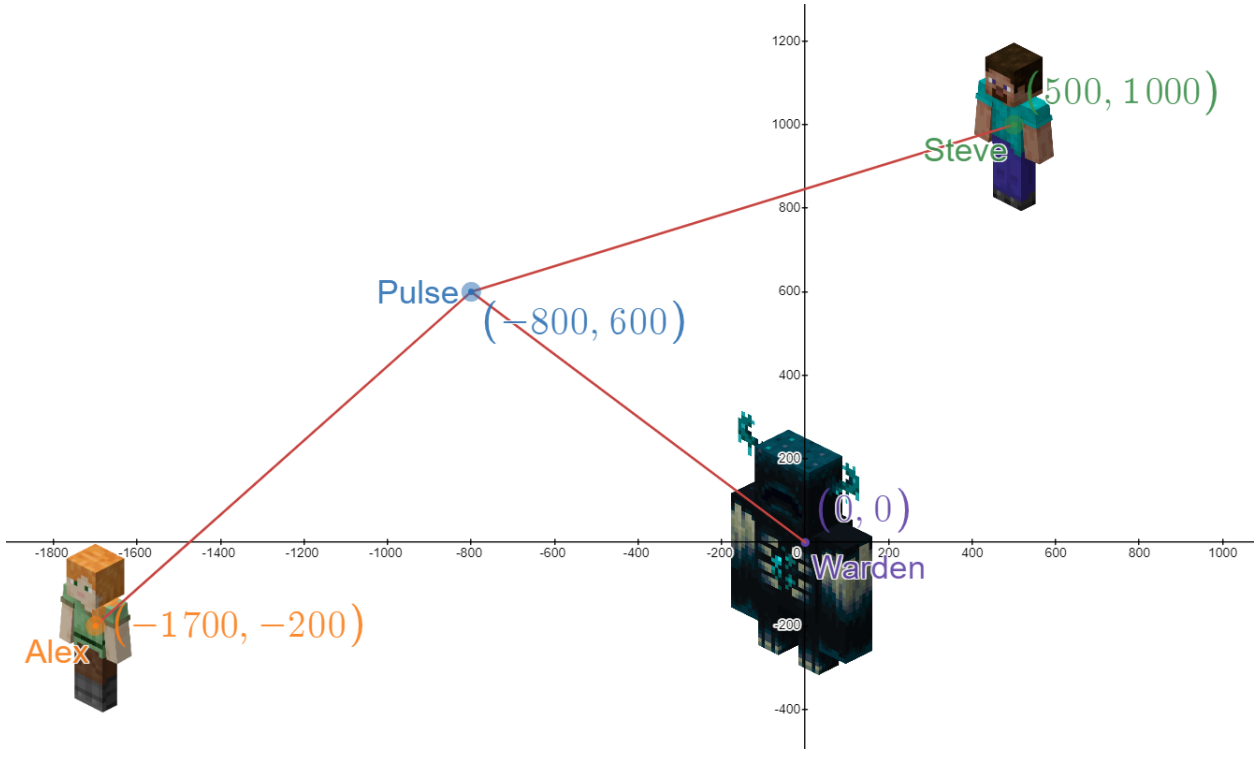
$$d((a, b), (c, d)) = \sqrt{(c - a)^2 + (d - b)^2}$$

After sending up to $\mathbf{P}$ *pulse* queries, you can send a *blast* query to blast at any two real number coordinates between **-10^6 and 10^6**. If the distance between a blast location and Steve's location and the distance between the other blast location and Alex's location are **both at most 100**, you successfully pass the test case. In other words, the condition is:

$$d((x_{b1}, y_{b1}), (\mathbf{X_S}, \mathbf{Y_S})) \le 100 \text{ AND } d((x_{b2}, y_{b2}), (\mathbf{X_A}, \mathbf{Y_A})) \le 100$$
$$\text{OR}$$
$$d((x_{b1}, y_{b1}), (\mathbf{X_A}, \mathbf{Y_A})) \le 100 \text{ AND } d((x_{b2}, y_{b2}), (\mathbf{X_S}, \mathbf{Y_S})) \le 100$$

If you are successful, the judge will respond with `CORRECT` and proceed to the next test case. If you are unsuccessful, the judge will respond with `WRONG_ANSWER` and your program should exit to receive a wrong answer verdict.

Here is a **new** Desmos graph for simulating and visualizing pulses and distance calculations!
Again, you can use it to try out examples, debug your code, do coordinate math, and more!



Interaction Format

This is an interactive problem! Unlike regular problems, your program and the judge will run simultaneously. Please see the contest guide for more information. Please flush your buffer as instructed by this post when you output, or use our template code that handles it for you. If you run into technical issues with interaction, please let us know with a clarification request!

Begin by reading a single line containing an integer $\mathbf{T}$ denoting the number of test cases that follow. For each test case:

1. Start by making up to $\mathbf{P}$ *pulse* queries. For each query:
   a. First, output a single line containing 3 space separated symbols `P` x_p y_p where:
      - The character `P` signals this is a *pulse* query.
      - The real numbers x_p y_p denote the coordinate of sculk to send this pulse to.
   b. Then, read a single line containing a non-negative real number d that denotes the rounded Euclidean distance of this pulse path.
2. Finish by making a single *blast* query as follows:
   a. First, output a single line containing 5 space separated symbols `B` x_{b1} y_{b1} x_{b2} y_{b2} where:
      - The character `B` denotes that this is a *blast* query
      - The real numbers x_{b1} y_{b1} x_{b2} y_{b2} denote the coordinates of the locations to blast.
   b. Then, read a single line containing a string that will be `CORRECT` or `WRONG_ANSWER`.
      - If the judge responds with `CORRECT`, you passed this test case.
      - If the judge responds with `WRONG_ANSWER`, your answer is incorrect, and your program should exit to receive a wrong answer verdict.

You can output the real numbers x_p, y_p, x_{b1} y_{b1} x_{b2}, and y_{b2} by expressing them in *decimal notation* like `123.456` or in *scientific notation* like `1.23456e+2` or `1.23456E2`. However, the pulse distance d will always be given in decimal notation, **rounded to 10^{-6}**.

If your program deviates from the interaction format (e.g. coordinate out of bounds, too many pulse queries, wrong number format, etc.), the judge will send `WRONG_ANSWER`, and your program should exit to receive a wrong answer verdict.

Constraints

Time Limit: **3 seconds** (I/O can be slow)

$1 \le \mathbf{T} \le 100$

$\mathbf{P} = 1500$

$-10^5 \le \mathbf{X_S}, \mathbf{Y_S}, \mathbf{X_A}, \mathbf{Y_A} \le 10^5$

$-10^6 \le x_p, y_p \le 10^6$

$-10^6 \le x_{b1}, y_{b1}, x_{b2}, y_{b2} \le 10^6$

Sample Interaction

The line spacing here is to emphasize the order in which interaction takes place only. Do not expect or output blank lines between each line of interaction.

Sample Input	Sample Output
3	
	P 100 200
5923.885760	
	P -800 600
7128.613018	
	P 1300 300.7
10876.748328	
	B 500.7 999.7 -1700.1 -200.1
CORRECT	
	P 12345.6789 98765.4321
846379.903725	
	B 69420 -42.1 1010 0.11
WRONG_ANSWER	

Sample Explanations

The judge begins by outputting `3`, the number of test cases. The judge also decides on Steve's and Alex's locations, ($\mathbf{X_S}$ = 500, $\mathbf{Y_S}$ = 1000) and ($\mathbf{X_A}$, $\mathbf{Y_A}$) = (-1700, -200).

The program begins by sending a *pulse* query at (x_p = `100`, y_p = `200`), to which the judge responds with the rounded distance of the pulse path to that sculk location: `5923.885760`.

Next, the program sends two more pulses at (x_p = `-800`, y_p = `600`) and (x_p = `1300`, y_p = `300.7`). The judge responds with `7128.613018` and `10876.748328`.

Finally, the program sends a *blast* query at (x_{b1} = `500.7`, y_{b1} = `900.7`) and (x_{b2} = `-1700.1`, y_{b2} = `-200.1`). These are not Steve's and Alex's exact locations, but d((x_{b1}, y_{b1}), ($\mathbf{X_S}$, $\mathbf{Y_S}$)) = d((500.7, 999.7), (500, 1000)) = 0.761577 and d((x_{b2}, y_{b2}), ($\mathbf{X_A}$, $\mathbf{Y_A}$)) = d((-1700.1, -200.1), (-1700, -200)) = 0.141421, which is close enough, so the judge responds with `CORRECT`.

The judge then decides on Steve's and Alex's locations for the next test case, ($\mathbf{X_S}$ = 61926, $\mathbf{Y_S}$ = -18290) and ($\mathbf{X_A}$ = -81928, $\mathbf{Y_A}$ = -73681).

The program makes a *pulse* query followed by a *blast* query that's too far away from Steve and Alex's locations. The judge responds with `WRONG_ANSWER`, and the program exits.