Problem 10: Steve was obliterated by a sonically-charged shriek
8 Point(s)

Problem ID: `warden`
Rank: `3`

Introduction

This is the first of a two part problem series! You can find the second part here.

You are the Warden—terror of the deep dark and sentinel of the ancient city. A trespassing player named Steve has entered your dominion, and you must destroy them at all costs! Although unable to see the intruder, you can track them by pulsing vibrations through sculk sensors throughout the city. Expose their location 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 $\mathbf{P}$ = 150 or fewer *pulse*s, find Steve and *blast* them to pass each test case.

The Warden is at (0, 0) on the 2D coordinate plane. Steve is at $(\mathbf{X_S}, \mathbf{Y_S})$, a real number coordinate between **-10^5 and 10^5** predetermined for each test case, but not given to you as input.

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

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

In other words, you will receive the value of:

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

where d is the distance function:

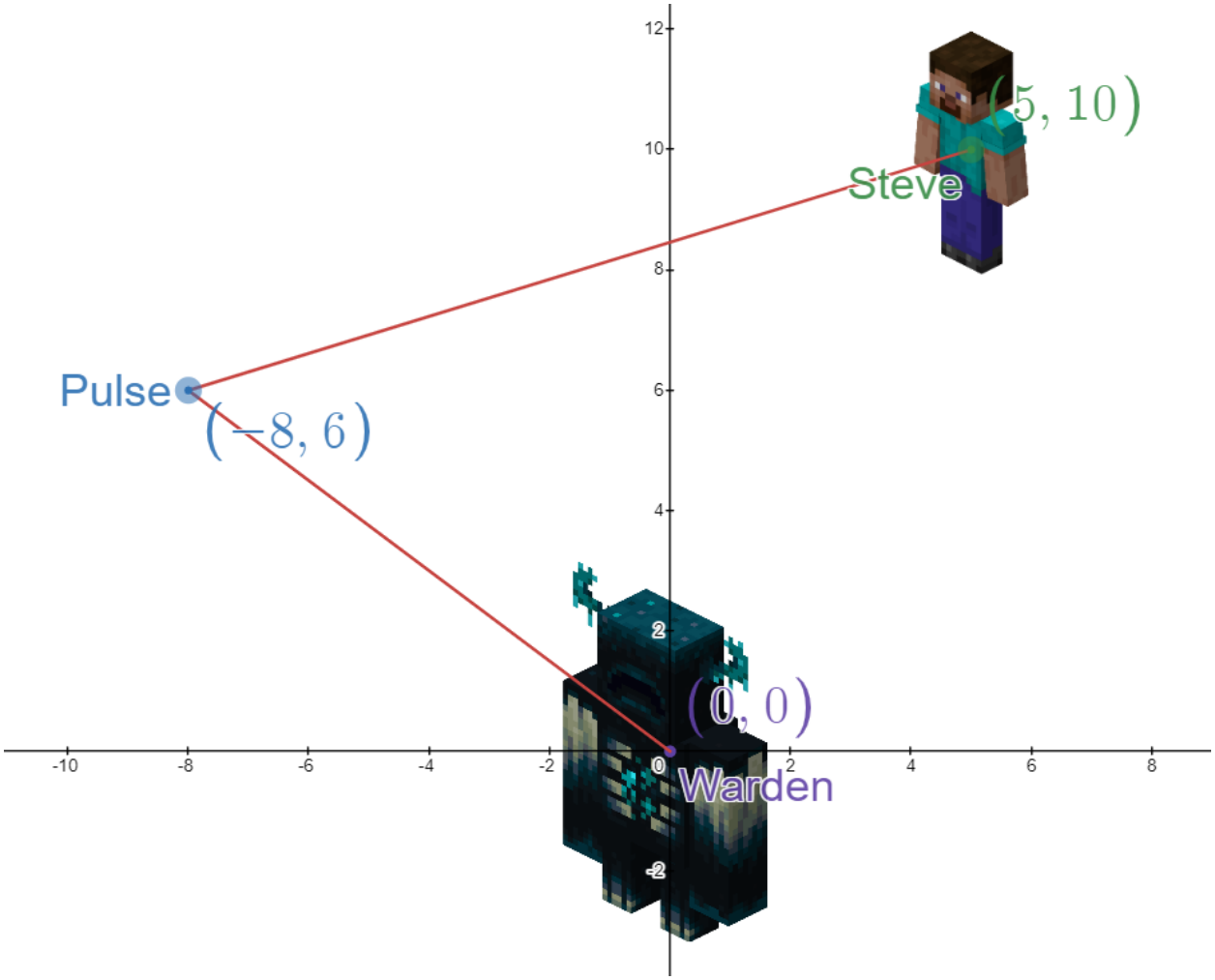
$$d((a, b), (c, d)) = \sqrt{(c - a)^2 + (d - b)^2}$$

After sending up to $\mathbf{P}$ *pulse* queries, you can send a *blast* query to any real number coordinate between **-10^6 and 10^6**. If the distance between your blast location and Steve's location is **at most 1**, you successfully pass the test case. In other words, the condition is:

$$d((x_b, y_b), (\mathbf{X_S}, \mathbf{Y_S})) \leq 100$$

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 Desmos graph for simulating and visualizing pulses and distance calculations! You can use it to try out examples, debug your code, do coordinate math, and more!



Note: a closed form mathematical solution exists, but we encourage you to seek a solution of a more computational nature. You may find it easier to generalize for the second part that way.

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 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 3 space separated symbols `B` x_b y_b where:
      - The character `B` denotes that this is a *blast* query
      - The real numbers x_b y_b denote the coordinate of the location 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_b, and y_b 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)

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

$\mathbf{P} = 150$

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

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

$-10^6 \le x_b, y_b \le 10^6$

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 1 2
22.360680	
	P -8 6
47.202941	
	P 13 3.7
47.398230	
	B 5.7 9.7
CORRECT	
	P 12345.6789 98765.4321
453313.504869	
	B 69420 -42.1
WRONG_ANSWER	

Sample Explanations

The judge begins by outputting `3`, the number of test cases. Before the first interaction, the judge also decides on Steve's location, ($\mathbf{X_S}$ = 5, $\mathbf{Y_S}$ = 10). This is not known by the program.

The program begins the first test case by sending a *pulse* query at (x_p = `1`, y_p = `2`). The judge responds with the rounded distance of the pulse path location: `22.360680`.

Next, the program sends two more pulses at (x_p = `-8`, y_p = `6`) and (x_p = `13`, y_p = `3.7`). The judge responds with `47.202941` and `47.398230`. Note that the program can send pulse locations that are non-integer or negative.

Finally, the program decides it's finished with pulsing even though it only used 3 out of the allowed $\mathbf{P}$ = 150 pulses. It then sends a *blast* query at (x_b = `5.7`, y_b = `9.7`). Although this is not exactly Steve's location of (5, 10), $d((x_b, y_b), (\mathbf{X_S}, \mathbf{Y_S}))$ = d((5.7, 9.7), (5, 10)) = 0.761577, which is close enough, so the judge responds with `CORRECT`.

The judge then decides on Steve's location for the next test case, ($\mathbf{X_S}$ = 61926, $\mathbf{Y_S}$ = -18290).

The program makes a *pulse* followed by a *blast* that's too far away from Steve's true location. The judge responds with `WRONG_ANSWER`, and the program exits **before the third test case**.