Back To School '19: Beautiful_Times' Fruit Scam

Time limit: 8.0s **Memory limit:** 256M

You are going to the farmers market to buy and sell fruit. There are N farmers markets that each buy and sell 2 types of fruit: Apples, and Bananas. Initially, each market will buy and sell each type of fruit at a fixed price decided by its farmer, who will keep these prices forever. Since the farmers are not sure about the prices of the other markets, **they will pick a price uniformly at random for each type of fruit**. Note that apples, and bananas may have different prices.

You want to try and figure out the prices of each of the 2 fruits for each market. However, since you are lazy, you will ask your friend Beautiful_Times to help you. You have unlimited apples, and bananas. Each day, you will first decide how many of each type of fruit to buy and sell. Then, you will ask Beautiful_Times to go to **each** market to buy and sell that amount of each fruit and collect the profit that you will make. If the price of the apple at market i is a_i, then you will gain a_i dollars for every apple you sell, and lose a_i dollars for every apple you buy. The same goes for bananas (b_i). The profit for that market is the total amount of money gained or lost for the 2 types of fruit.

Unfortunately, Beautiful_Times has poor memory and he will only remember one value: whether there were more markets where you made a profit of strictly greater than P_j dollars (P_j can be negative) than markets where you made a profit of strictly less than P_j dollars.

On some days, instead of asking Beautiful_Times to go to all markets, you will ask him to compare the results of certain previous days. You will choose two days j and k, and he will give you one piece of information:

- Considering only the markets where you earned a profit of *at least* P_j on day j, **and** a profit of *at least* P_k on day k, choose two of the markets uniformly at random and output the price of the banana. If there is only one market, Beautiful_Times will output `-1` for the second value. If there are no such markets, he will output `-1` twice.

However, Beautiful_Times does not allow you to choose any two days. He will only allow you to compare day k with day j **if** the result from day j was that there were an equal number of markets with a profit greater than P_j than there was with a profit less than P_j.

Can you use this information to determine the prices of each of the 2 fruits for all markets?

There will be T test cases. You only need to output the correct answer for at least 90% of the cases to be accepted.

Interaction

The first line for each case will be the integer N, the number of markets.

Next, you must start interaction with Beautiful_Times.

Output `? A_b A_s B_b B_s P_j` to ask Beautiful_Times to buy A_b apples, B_b bananas, and sell A_s apples, B_s bananas, at each market. $0 \leq A_b, A_s, B_b, B_s, |P_j| \leq 10^{18}$. If there were more markets where you made a profit of strictly greater than P_j dollars than strictly less than P_j dollars, then you will receive the integer `1`. If there were fewer markets, you will receive the integer `-1`. If they are equal, you will receive the integer `0`.

Output `C j k` to compare day j and day k. You may only do this if on day j, you asked a question where the result was that there were an equal number of markets where the profit was greater than P_j as there were less than P_j. You will receive two integers `x_1 x_2`, the prices of the banana at two markets chosen uniformly at random from the markets where you earned a profit of at least P_j on day j, **and** a profit of at least P_k on day k. If there is only one market, Beautiful_Times will output `-1` for the second value. If there are no such markets, he will output `-1` twice.

When you have determined the prices of each market, output `!`. This should be followed by N lines in the form `a_i b_i`, meaning the i^{th} market buys and sells apples at a_i dollars, and bananas at b_i dollars. The markets can be printed in any order.

You must then continue onto the next case. *The interactor will not output anything to indicate the end of this case, or whether your answer is correct or not.*

If at any point, you receive a `-400`, this means Beautiful_Times could not parse your question/answer or the question was invalid, and you must exit your program immediately to ensure you receive `WA`. Otherwise, *your verdict could be undefined*.

REMINDER: After each output, remember to *flush*. In C++, this can be done with `fflush(stdout)` or `cout << flush` (depending on whether you use `printf` or `cout`). In Java, this can be done with `System.out.flush()`. In Python, you can use `sys.stdout.flush()`.

Note: The interactor may use up to 1 second of the time limit.

Constraints

For all subtasks:

T will always be equal to 10. This means you only need to output the correct answer for at least 9 cases to be accepted.

The number of markets will be between 1 and 80 $(1 \leq N \leq 80)$. Each farmer will set the prices to be between 0 and 10^7 dollars. Formally, $0 \leq a_i, b_i \leq 10^7$.

You must determine each market's prices in at most $40\,000$ days. Each day, you may ask at most 1 question. Outputting the answer does not count as a question. However, comparing 2 days or asking Beautiful_Times to go to the markets both count as a question.

Subtask 1 [5%]

$N = 1$

Subtask 2 [10%]

$N \leq 2$

Subtask 3 [10%]

$N \leq 4$

Subtask 4 [25%]

$N \leq 10$

Subtask 5 [50%]

No additional constraints.

Sample Interaction

Please note that the sample input does not satisfy all of the constraints for some of the subtasks. It will not appear in any of the test cases.

In this sample, $T = 2$. Remember that $T = 10$ in the actual test cases.

There is 1 market in the first test case.
Market 1 buys and sells apples at $\$8$, and bananas at $\$10$.

There are 3 markets in the first test cases.
Market 1 buys and sells apples at $\$2$, and bananas at $\$3$.
Market 2 buys and sells apples at $\$6$, and bananas at $\$1$.
Market 3 buys and sells apples at $\$5$, and bananas at $\$4$.

`>>>` denotes your output. Do not print this out.

```
1
>>> ? 6 3 4 9 30
-1
>>> !
>>> 3 7
3
>>> ? 1 2 5 3 -3
0
>>> ? 1 4 2 9 26
1
>>> C 1 2
4 -1
>>> !
>>> 5 4
>>> 2 3
>>> 6 1
```

Sample Explanation

In the first test case, Beautiful_Times collects a profit of $\$26$ at market 1 on day 1. Since this is less than $\$30$, he responds with `-1`. An answer is then printed. Although it is incorrect, the judge will continue on with the next test cases.

In the second test case, Beautiful_Times collects profits of $-\$4, \$4, -\$3$ on day 1 at markets $1, 2, 3$ respectively. Since there are an equal amount of markets that made a profit greater than $-\$3$ and less than $-\$3$, he responds with `0`. On day 2, Beautiful_Times collects profits of $\$27, \$25, \$43$. Since there are more markets that made a profit greater than $\$26$ than markets that made a profit of less than $\$26$, he responds with `1`. On day 3, days 1 and days 2 are compared. This comparison is valid since on day 1; Beautiful_Times responded with `0`. Since market 3 made a profit of at least $-\$3$ on day 1 and $\$26$ on day 2, and no other markets fit this criteria, Beautiful_Times responds with `4 -1`. Finally, the correct answer is printed. Beautiful_Times will not respond after the last test case. Note that the markets may be printed in any order.

Sample Interactor

You are given two Python scripts to test your solution locally. **You must have Python 3 installed in order to run the files.** Please notes that the sample interactor may behave differently than the actual interactor.

There are 2 files: `interactive_runner.py` and `interactor.py`. To use the interactor, run the following command:

```
python3 interactive_runner.py python3 interactor.py SUBTASK_NUMBER -- CMD_LINE_SOLUTION
```

`SUBTASK_NUMBER` is an integer between 1 and 5 corresponding to the subtask you would like to test.

`CMD_LINE_SOLUTION` is the command required to run the solution. In C and C++, the command is `./binary_executable_name`. In Java, the command is `java class_name`. In Python 3, the command is `python3 file_name.py`.

The interactor will run 10 test cases with random values of N. The test cases may vary between runs. You are encouraged to test on your own, as well as modify the interactor if you wish.

Attachment - interactor.py
Attachment - interactive_runner.py