Problem 3: least addicted genshin player
4 Points

Problem ID: `genshin`
Rank: `1`

Introduction

It's 4/12 and Genshin Impact 3.6 is here, bringing to the game a new character: Baizhu!

Ever since he was announced as a playable character 2.5 years ago, I haven't been able to stop thinking about my one and only Bazihu. The way his luscious emerald hair flows in the wind, the way his gorgeous golden eyes glitter in the sunlight, the way his enchanting voice touches my heart, all this I find irresistible. I need to get him as soon as he's released.

There's just one small problem. Genshin has a gacha system where players spend primogems for a random chance at getting characters and items. I need to buy more primogems (with real life money), but my crippling gacha addiction has completely drained my wallet. I must secretly use my mom's credit card instead. But I don't want her to notice the charges, so I will only make purchases $1 at a time. It's fine; she probably won't mind. She'd understand… right?

I'm not sure why people are calling me a simp for being so dedicated to Baizhu. I just really want to add him to my collection. I know I can't stop thinking about him but it's not like I'm obsessing over him or anything. I'm just really really excited, that's all.

Problem Statement

This is an interactive problem! Starting with $\mathbf{P}$ *primogems*, get Baizhu by making *wish* queries and *buy* queries without *buying* more *primogems* than necessary.

Each interaction, you can perform one of the two following queries. You may query as many times as necessary until you get Baizhu.

- *Buy* primogems:
  - Get more *primogems* for wishes in case the amount you start with is not enough.
  - You always get 60 *primogems* when you *buy*.
- *Wish* to get a random item:
  - Each *wish* costs 160 *primogems*. You can only wish if you have at least 160.

- Each *wish* has a ~0.6% chance of getting Baizhu.
- You are guaranteed to get Baizhu within 180 wishes.

The sequence of *wish* items is random but pre-determined. When you get Baizhu, if you bought more primogems than necessary (in other words, you could have gotten him by *buying* one less time), you will get a `WRONG_ANSWER` verdict. Otherwise, you will get a `CORRECT` verdict.

*Note: This problem—alongside **all other problems in this contest**—has templates available in Python, Java, and C++! You can find them in the contest.zip provided at the start of the contest. The template for this problem implements interaction for you as two simple functions that you can call from within your solution.*

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 reading a single line containing an integer $\mathbf{P}$, the number of starting *primogems*.
2. Then, make *buy* queries or *wish* queries in any order until you get Baizhu. You can only make *wish* queries if you have at least 160 *primogems*.
   a. To make a *buy* query:
      i. First, output a single line containing the word `buy`
      ii. Then, read a single line that contains the following:
      `you got: 60 primogems`
   b. To make a *wish* query:
      i. First, output a single line containing the word `wish`
      ii. Then, read a single line in the following format:
      `you got:` i
      where i is the name of the item you just got from wishing.
      - You get Baizhu if the name of the item is `baizhu`
3. Finally, read a single line containing `CORRECT` or `WRONG_ANSWER`

If at any point your program deviates from the interaction format (e.g. invalid query type, wishing with insufficient *primogems*), the judge will send `WRONG_ANSWER`. If your program reads `WRONG_ANSWER` at any point, you should exit to receive a wrong answer verdict.

Constraints

Time Limit: **2 seconds**

Item names contain only lowercase alphabet letters `abcdefghijklmnopqrstuvwxyz` and underscores _ , and will not exceed 40 characters. A full list of item names can be found here.

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

$0 \le \mathbf{P} \le 10^5$

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	
420	
	wish
you got: favonius_sword	
	wish
you got: ningguang	
	buy
you got: 60 primogems	
	wish
you got: baizhu	
CORRECT	
1337	
	buy
you got: 60 primogems	
	wish
you got: baizhu	
WRONG_ANSWER	

Sample Explanations

The judge begins by outputting `3`, the number of test cases.

For test case #1, the program begins by sending a *wish* query, spending 160 *primogems*, and getting `favonius_sword`, which is not `baizhu`. The program now has 420 - 160 = 260 *primogems*. Then, the program sends another *wish* query, getting `ningguang`, and is now left with 260 - 160 = 100 *primogems*. The program then sends a *buy* query and now has 100 + 60 = 160 *primogems*. Finally, the program sends another wish query and gets `baizhu`. Since every *buy* query was necessary, the judge responds with `CORRECT`, and the program passes the first test case.

For test case #2, the program begins with buying, and now has 1337 + 60 = 1397 *primogems*. The program then sends a *wish* query and gets `baizhu`. The *buy* query was not necessary, as the program could have just *wished* instead of *buying* on the first query. Thus, the judge responds with `WRONG_ANSWER`, and both programs terminate.

Although this test file has 3 test cases, because both programs terminate before the third one, the judge never gets to the final test case.