The Hanged Man (XII)

Hangman is a pen-and-paper word game usually played between two players. Let's label one player as **Judge** and the other as **Player**. The game proceeds as follows,

- **Judge** secretly selects a word, and writes down one blank for each letter in the word.
- **Player** says an English letter, and **Judge** reveals which positions in their secretly chosen word contain that letter (if any). This is done by just writing that letter in the corresponding blanks where it should go.
- **Player** keeps saying letters until they can guess what the secret word is.

The objective is usually for **Player** to guess the secret word in as few "misses" as possible, usually tracked by drawing more and more parts of a stick figure "hanged man" on a gallows.

Your task is to write a program that functions as the **Player**, and aims to guess the secret word with as few misses as possible.

To simplify the game, we make the following assumptions.

- **Judge** is honest and will not cheat. It will not secretly change its chosen word during the game.
- **Judge** chooses a single valid English word. A word is valid if it belongs to this[1] word list. Note that the only characters in a valid word are the 26 lowercase English letters.

[1] `https://github.com/dwyl/english-words/blob/master/words_alpha.txt`

Interaction

Your program will communicate with the judge by reading from standard input and writing to standard output (so the usual methods of I/O used in comp prog). Everything the judge says goes to standard input, and everything you wish to say should go to standard output.

First, the judge outputs a line containing a single integer T, the number of test cases. Each test case then proceeds as follows.

First, the judge outputs a line containing some number of asterisks, representing their secret word.

Then, you have two options. Output a line containing one of the following.

- `ASK` c

  Here, c is a lowercase English letter.

  If c appears in the secret word, then the judge replaces the asterisks with c at each position where it would appear in the word. Then, it responds with a single line containing their secret word with the newly revealed letters.

  If c does not appear in the string **or** you have already said c in a past `ASK` query, the judge responds with a single line containing `MISS` instead.

- `ANSWER` s

  Here, s is a string of lowercase English letters.

  If s is equal to their hidden word, the judge responds with a single line containing `GOOD`, then moves on to the next test case.

  If s is **not** equal to their hidden word, the judge responds with `MISS`.

This repeats until you have successfully guessed the secret word.

The penalty of your program is simply the average number of `MISS` responses it incurred, over all test cases. The goal is to guess all the words while minimizing this penalty.

After printing a query, do not forget to output an endline and flush the output. To do this, use:

- `cout.flush()` in C++
- `sys.stdout.flush()` in Python (don't forget to `import sys`)
- See the documentation for other languages.

Example Interaction

```
JUDGE           PLAYER
2
****
                ASK a
**a*
                ASK e
MISS
                ASK s
MISS
                ASK l
*la*
                ASK y
*lay
                ANSWER clay
MISS
                ANSWER play
GOOD
*****
                ASK e
***ee
                ASK r
**ree
                ASK s
MISS
                ASK t
t*ree
                ASK h
three
                ANSWER three
GOOD
```

Constraints

Your program will be tested against $T = 500$ words. The breakdown is as follows,

- 200 valid words randomly sampled from the word list.
- 200 valid words randomly sampled from a natural language text.
- 100 valid words specially chosen by the coach.

The test cases will not necessarily be given in that order. The actual word list will not be publicly released until after the results are announced.

Scoring

Submit your program to the Google Classroom assignment on or before November 27 (three weeks from November 6) to have it included in the competition. I will run all programs **once** locally on my machine, and this will form the basis of the final scores on the leaderboard.

My laptop has a `1.8 GHz` processor speed and `8 GB` of RAM.

Programs will be ranked according to the average number of misses per word across all test cases; **less** misses means placing higher on the leaderboard.

External Files

You are reassured that the word list `words_alpha.txt` exists in the same working directory as your program when it is run. If your program relies on other external files, then submit those as well (within reason).

Time Limit

If your program takes more than 5 minutes to terminate on my machine, it will be given a penalty of $+\infty$.

Testing

I coded a Judge program in Java which you can download here [2] that will aid you in your testing. It will be the one to manage the interactions between your program and the judge. This is also the program I will use to judge it.

You need a Java Runtime Environment of 8 or higher in order to run it. The `README` contains the details on how to use it.

Please check the Discord or the Github for announcements because I will probably be frequently updating the interactor to fix bugs, etc.

[2] https://github.com/gfmortega/hangman-judge