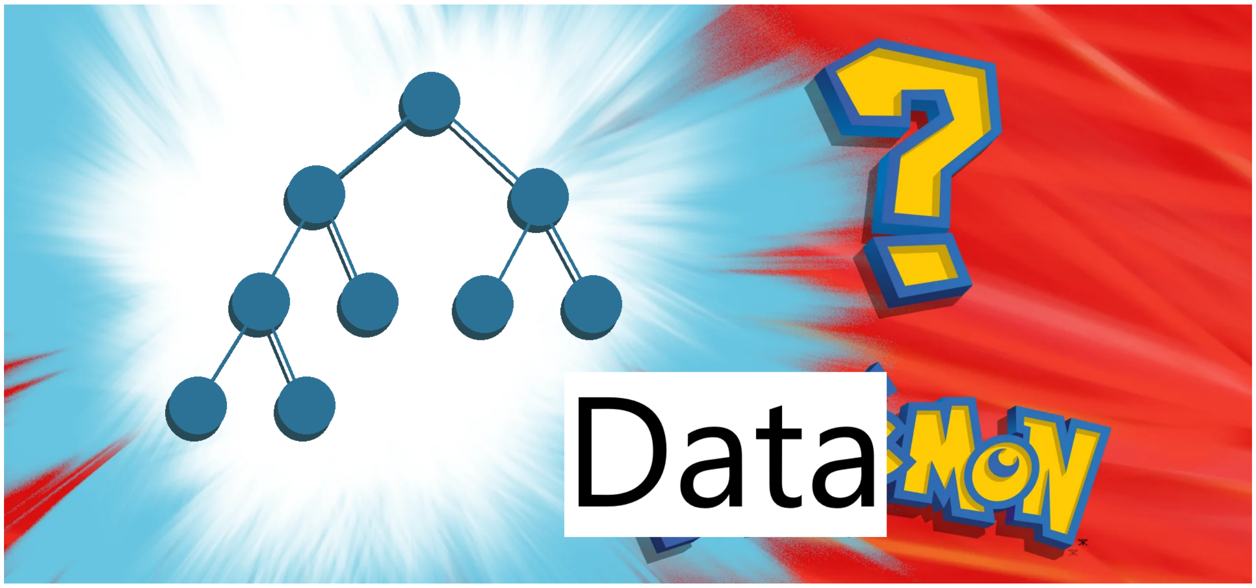
Problem 4: Who's that Datamon?
4 Points

Problem ID: `wtds`
Rank: `1`

Introduction

You, the intrepid trainer, have just caught your first Datamon! It wasn't easy, but your deft evasion of petting zoo officials has rewarded you with a Datamon to call your very own. However, as you begin to fill out the paperwork to legally register the Datamon under your name, you're stumped by the first question: Who's that Datamon?



Unfortunately, ever since Professor Ben confiscated and sold your phone on the dark web, you've been unable to watch blaring off-brand Subway Surfers gameplay at full volume during Datamon Recognition class. This violation of your rights has rendered you hopelessly unable to pay attention to lectures—how will you tell who's that Datamon now?

Thankfully, after discovering Professor Ben's diary inside their car, you've discovered that they have an intense personal interest in studying the bowel movements of Datamon in their free time. Hopefully, your limited reading ability will allow you to glean enough information from their diary to help you identify your Datamon!

Problem Statement

Given a Datamon of unknown species in your possession, you want to correctly identify it by feeding it numbers and watching it poop. There are three possible species and you have the following information about their stomachs:

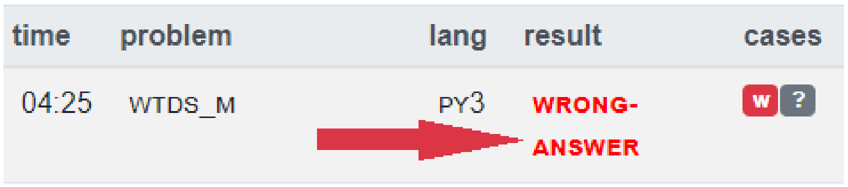
- `queueon`: Always poops numbers in the order that it was fed, first in first out (imagine people lining up in a bread line, first come first serve).
- `stackeon`: Always poops the most recent number that it was fed, first in last out (imagine a stack of plates; you can only remove or stack on the top plate).
- `heapeon`: Always poops the number with the maximum value (imagine an American politician who prioritizes meetings with highest-paying lobbyists first).

The Datamon begins with an empty stomach. During each interaction, you can perform one of the three following queries. You may query as many times as necessary until you identify the Datamon.

- *Feed*: feed a number to the Datamon:
  - The number gets added to the Datamon's stomach.
  - The judge responds with `OK`
- *Poop*: get the Datamon to poop out a number:
  - The number that gets pooped out depends on the species as described above.
  - The pooped number is then removed from the Datamon's stomach.
  - The judge responds with the number pooped out.
  - The Datamon is unable to poop if there are no numbers inside its stomach to poop out.
- *Guess*: guess the species of the Datamon:
  - If your guess is correct, the judge responds `CORRECT` and the next test case begins.
  - If your guess is wrong, the judge responds `WRONG_ANSWER` and ends the interaction.

*Note: Templates are available for this problem—and **all other problems in this contest**—in Python, Java, and C++! Find them in the contest.zip provided at the start of the contest. Templates handle input and output for you, so you can just fill out a single function!*

*Note: If you make an incorrect submission, you can **click your result** to see a more detailed explanation of what went wrong and an interaction log between your program and the judge:*



time	problem	lang	result	cases
04:25	WTDS_M	PY3	WRONG-ANSWER	W ?

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 a single integer $\mathbf{T}$ denoting the number of test cases that follow. For each test case:

A. Start by making *feed* and *poop* queries in any order.
   - To make a *feed* query:
     i. First, output a single line in the following format:
        `feed` i
        where i is an integer denoting the number to feed the Datamon.
     ii. Then, read a single line containing the string `OK`
   - To make a *poop* query:
     i. First, output a single line containing the string `poop`
     ii. Then, read a single line containing the integer pooped out by the Datamon.

B. To finish, make a *guess query.*
   - To make a *guess* query:
     i. First, output a single line in the following format:
        `guess` s
        Where s is a string denoting your guess of the species of the unknown Datamon, selected from the choices of `queueon`, `stackeon`, or `heapeon`
     ii. Then, read a single line containing either `CORRECT` or `WRONG_ANSWER`
        - If `CORRECT`, the next test case begins.
        - If `WRONG_ANSWER`, your program should exit.

If at any point your program deviates from the interaction format (e.g. invalid query, making a *poop* query when its stomach is empty, etc.), 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

$\mathbf{T} = 100$

For all feed queries, the number i being fed must be an integer and $0 \leq i \leq 100$.

The only three possible species are `queueon`, `stackeon`, or `heapeon`

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.

```
Judge Output   Contestant Output

         100
               feed 5
          OK
               feed 7
          OK
               poop
           5
               feed 6
          OK
               poop
           7
               guess queueon
     CORRECT
               feed 5
          OK
               feed 5
          OK
               feed 5
          OK
               poop
           5
               poop
           5
               guess heapeon
     CORRECT
               feed 9
          OK
               feed 7
          OK
               feed 15
          OK
               poop
          15
               guess stackeon
WRONG_ANSWER
```

Sample Explanations

The judge begins by outputting `100`, the number of test cases.

For test case #1, the program begins by sending a *feed* query, which feeds the number 5 to the Datamon and the judge outputs `OK`. The program then sends another *feed* query, with 7. The next query is a *poop* query, after which the Datamon poops out 5. Then, the program sends another *feed* query with 6. After that, when the *poop* query is sent, 7 is pooped. Finally, the program makes a *guess* query that the Datamon, in this case, is a queueon, and the judge gives a `CORRECT` verdict. No other Datamon species could have emulated the same behavior given this sequence of queries.

For test case #2, note that the *feed* query can be used to feed duplicate numbers. Specifically, the Datamon is fed the number 5 three times consecutively. Following this, the program sends two *poop* queries, both of which poop out 5. The program then *guesses* that the Datamon is a `heapeon`, and the verdict is `CORRECT`. While the program happened to correctly guess heapeon, a stackeon or queueon would have also exhibited the same behavior given this series of queries.

For test case #3, the program begins by sending three *feed* queries, feeding 9, 7, and 15 to the Datamon. The next query sent in the program is *poop*, after which 15 is pooped from the Datamon. The program finally *guesses* that the Datamon is a `stackeon`, and the judge gives a `WRONG_ANSWER` verdict—after which the program exits. While the behavior observed aligns with that of a stackeon, the behavior also aligns with that of a heapeon, which happened to be the actual Datamon species being interacted with.

While there were 100 test cases, the interaction ended after the third test case due to receiving a `WRONG_ANSWER` verdict.

This page intentionally left blank.