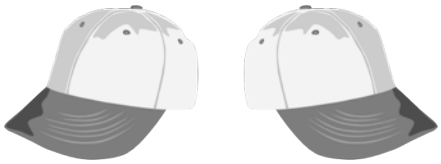
1. You Will All Conform

When people are free to do as they please, they usually imitate each other. – Eric Hoffer.

Programming constructs and algorithmic paradigms covered in this puzzle: Lists, tuples, functions, control flow including if statements and for loops, and print statements.

Let's say we have a whole bunch of people in line waiting to see a baseball game. They are all hometown fans and each person has a team cap and is wearing it. However, not all of them are wearing the caps in the same way – some are wearing their caps in the normal way, and some are wearing them backwards.

People have different definitions of normal and backwards but you, the gatekeeper thinks that the cap on the left below is being normally worn and the one on the right is being worn backwards.



You are the gatekeeper and can only let them in to the stadium if the entire group has their caps on in the same way – either all forwards or all backwards. Because everyone has different definitions of forward and backward, you can't tell them to wear their cap forwards or backwards. You can only tell them to flip their caps. The good news is that each person knows what position they are in the line, starting with the first person having position 0 and the last one position n – 1. You can say things like:

> *Person in position i please flip your cap.*
> *People in positions i through j (inclusive) please flip your caps.*

What you would like to do is to minimize the number of requests you have to shout out to save your voice. Here is an example:

We have 13 people standing in line in positions 0 through 12 inclusive. Since there are six people with caps on forwards, we could shriek out 6 commands, e.g.,

Person in position 0 please flip your cap.

And repeat for person in positions 1, 5, 9, 10 and 12. A voice-saving measure would exploit the second type of command and only yell out four commands:

People in positions 0 through 1 please flip your caps.
Person in position 5 please flip your cap.
People in positions 9 through 10 please flip your caps.
Person in position 12 please flip your cap.

And this will get everyone to have caps on backwards.

However, in this example, we can do one better. If we scream out:

People in positions 2 through 4 please flip your caps.
People in positions 6 through 8 please flip your caps.
Person in position 11 please flip your cap.

This will get everyone with caps on forwards.

How can we generate a minimum number of commands? A harder question: Can you think of a way of generating the commands as you walk down the line for the first time?

Exercises

Exercise 1: One slightly annoying thing is that executing `pleaseConform(cap1)` prints:

```
People in positions 2 through 4 flip your caps!
People in positions 6 through 8 flip your caps!
People in positions 11 through 11 flip your caps!
```

It should print for the last command:

```
Person at position 11 flip your cap!
```

Modify the code so the commands sound more natural.

Exercise 2: Modify `pleaseConformOnepass` to print more natural commands as in Exercise 1, and ensure that it does not crash on an empty list.

Hint: You will need to remember the beginning of the interval (and not print on Line 6).

Puzzle Exercise 3: Suppose there are bareheaded people in the line. We'll represent them with the `'H'` character. So for example we might have:

```
cap3 = ['F','F','B','H','B','F','B',
        'B','B','F','H','F','F']
```

We don’t want to confuse the bareheaded people by telling them to flip their non-existent caps and perhaps cause one of them to try to steal a cap from the person ahead in line. Therefore, we want to skip over all the `'H'` positions. Modify `pleaseConform` so it generates a correct and minimal set of commands. For the above example it should generate:

```
Person in position 2 flip your cap!
Person in position 4 flip your cap!
People in positions 6 through 8 flip your caps!
```

6.S095 Programming for the Puzzled
January IAP 2018