Exercises I/O

1 Hello, World!

- Write a Hello World program that prints some text.
- Write a program that asks for your name, reads it from stdin and greets that person.
- Write a program that counts the number of lines read from stdin
- Write a program that transforms text from stdin to upper case.

2 Scoreboard

Make a program that calculates the scores of some players in the following way. Input is given in a file, starting with a line stating which players were playing, each represented as an uppercase letter. The next line shows, the order in which players gained or lost points as a string, where an uppercase letter means the corresponding player gained a point and a lowercase letter means that player lost a point.

As an example two players `A` and `B` played a short game:

```
AB
AABaB
```

First `A` won two rounds, then `B` won, then `A` lost, and finally `B` won again. The final result is `A` has a score of `1` and `B` a score of `2`.

You should write the result to a new file, in a reasonable way. For example each player on its own line:

```
A: 1
B: 2
```

But initially you should probably just print/show the object, as I prefer you thinking about how to solve the problem, rather than details of how to present it in a pretty way.

Hint: Start with a simple solution that works according to some (or a lot) assumptions, and later refactor your solution to be better.

Hint: You might find that `Data.Map.Strict` supplies a helpful data structure and helper functions.

MORE EXAMPLES

Input:

```
ABC
ABCABCbaa
```

Output:

```
A: 0
B: 1
C: 2
```

Input:

```
quickfox
QFOqUQFIIOoxXCckxxoQ
```

Output:

```
C: 0
F: 2
I: 2
K: -1
O: 0
Q: 2
U: 1
X: -2
```

2.1 Some improvements

After you made your first version, improve it, so it works from fewer assumptions. This may be fast if you made a great solution initially.

Things you may consider improving:

- If you assumed the players came in ordered like `['A'..]`, you could change it so it works for any order of players, such as `"BXM"`, or even `"bXm"`.
- Improve the printing of the result, so it shows more like the example.

3 Lazy IO

Make two programs, one that gets the contents of a file, and prints the first character, and another one that prints the last character of that file.

Use some kind of time tool, and see the difference in time when running the programs on a large file, such as the `lipsum.txt` file uploaded here.

4 Another game: Nim

The game Nim exists in many variants, usually between two players. One of the simple variants is played with a pile of items, say, 15, and each round the current player must take a number of items in some range, say, 1-4 items. The goal is to not take the last item, so the range does not include 0.

You should implement this game, you can decide on the rules to play by, or let them be decided at the beginning of the game. The program should write the number of items left after each player has had their turn.

To save some time, you don't need to think about which players turn it is, the people playing it can keep track of that themselves.

4.1 Improvements

Ideas for additions to the game:

- The program actually terminates when the game is finished (number of items == 0), with a small message.
- As mentioned, the number of items and the range is decided by input at the beginning of the program.
- Make it so there are multiple piles each with their own number of items, so the player first chooses which pile to draw from, then the number of items to draw.
- Make a datastructure for the game, and make it a custom instantiation of the `Show` typeclass. Use this to show the game, rather than simply the number of items left in the pile.
- Make it part of the `Game` typeclass, from the typeclass exercise from sheet 3, if you made it.