Specs

Write a program that prints the lyrics of the song 99 Bottles of Beer.

It will take as an argument the starting number of bottles, given through the command line when executing the program. This program should work this way.

```
$ ruby lib/beer_song.rb 5
5 bottles of beer on the wall, 5 bottles of beer!
Take one down, pass it around, 4 bottles of beer on the wall!
4 bottles of beer on the wall, 4 bottles of beer!
Take one down, pass it around, 3 bottles of beer on the wall!
3 bottles of beer on the wall, 3 bottles of beer!
Take one down, pass it around, 2 bottles of beer on the wall!
2 bottles of beer on the wall, 2 bottles of beer!
Take one down, pass it around, 1 bottle of beer on the wall!
1 bottle of beer on the wall, 1 bottle of beer!
Take one down, pass it around, no more bottles of beer on the wall!
```

Hint

- You can communicate arguments to your program from the command line using ARGV
- I hope you noticed the change from bottl**es** to bottl**e** when only 1 bottle remains !

A bit of context on ARGV

Any Ruby program you write runs inside another piece of software: the Ruby interpreter. And that interpreter is itself running inside another piece of software: your operating system. These software layers are called the environment and there are many ways you can exchange data between the environment and your program.

One way is via the ARGV constant which comes pre-defined in every Ruby program. It is an Array of Strings representing the command line arguments. Consider this simple program

testing_argv.rb

```
puts "*** Command line arguments ***"
puts ARGV.inspect
````
Now run it in the terminal this ways
```

$ ruby testing_argv.rb un deux trois *** Command line arguments *** ["un", "deux", "trois"] $ ruby testing_argv.rb "un et deux" trois *** Command line arguments *** ["un et deux", "trois"] ```
```