**Grade School**

In Practice Mode

Introduction

Given students' names along with the grade that they are in, create a roster for the school.

In the end, you should be able to:

* Add a student's name to the roster for a grade
  + "Add Jim to grade 2."
  + "OK."
* Get a list of all students enrolled in a grade
  + "Which students are in grade 2?"
  + "We've only got Jim just now."
* Get a sorted list of all students in all grades. Grades should sort as 1, 2, 3, etc., and students within a grade should be sorted alphabetically by name.
  + "Who all is enrolled in school right now?"
  + "Let me think. We have Anna, Barb, and Charlie in grade 1, Alex, Peter, and Zoe in grade 2 and Jim in grade 5. So the answer is: Anna, Barb, Charlie, Alex, Peter, Zoe and Jim"

Note that all our students only have one name. (It's a small town, what do you want?)

For bonus points

Did you get the tests passing and the code clean? If you want to, these are some additional things you could try:

* If you're working in a language with mutable data structures and your implementation allows outside code to mutate the school's internal DB directly, see if you can prevent this. Feel free to introduce additional tests.

Then please share your thoughts in a comment on the submission. Did this experiment make the code better? Worse? Did you learn anything from it?

Getting Started

Make sure you have read the "Guides" section of the [C track](https://exercism.io/my/tracks/c) on the Exercism site. This covers the basic information on setting up the development environment expected by the exercises.

Passing the Tests

Get the first test compiling, linking and passing by following the [three rules of test-driven development](http://butunclebob.com/ArticleS.UncleBob.TheThreeRulesOfTdd).

The included makefile can be used to create and run the tests using the test task.

make test

Create just the functions you need to satisfy any compiler errors and get the test to fail. Then write just enough code to get the test to pass. Once you've done that, move onto the next test.

As you progress through the tests, take the time to refactor your implementation for readability and expressiveness and then go on to the next test.

Try to use standard C99 facilities in preference to writing your own low-level algorithms or facilities by hand.