Making an Online Quiz

It's time to study for a test. One of the best ways to do so is to come up with your own questions and to quiz yourself. We can write a computer program to manage the quiz. It will interactively ask the user each question, and at the end report how many questions were answered correctly.

Ideally, it should not be necessary to change the program if all you want to do is to change the questions and answers to the quiz. Therefore, the questions and answers should not be in the source code. Where else could they be? We will keep them in a text file that will be used as input to your program.

To illustrate how the program should work, consider the following illustration of an I/O session:

```
The quiz will now begin.  There are 3 questions.

Which planet's orbit lies between those of Jupiter and Uranus?  Neptune
Sorry, that answer is not correct.  The correct answer is Saturn.

How many moons does Venus have?  Enter a number.  0
Great!  That is the correct answer.

Mars has two moons.  One is Deimos.  Name the other one.  Ceres
Sorry, that answer is not correct.  The correct answer is Phobos.

The quiz is over.  You answered 1 question correctly.
```

Behind the scenes, the program had already read in and stored the questions and answers in the computer's memory.

Implementation:

1. First, you need to create the input file. Create a text file called input.txt, and enter your desired questions, one per line. Next, insert two blank lines after each question. Now, type in the correct answer directly underneath each question. Leave the second blank line alone, so that you can visually separate the questions. For example, if you have 3 questions, then the questions themselves will appear on lines 1, 4, and 7, the answers will appear on lines 2, 5, and 8, and lines 3, 6, and 9 will be empty.

2. The first thing that your computer program needs to do is to read in input.txt. It would be a good idea to know right away how many questions there are. So, count the number of lines in the input file. If you divide by 3, you should get the number of questions. Test your code, and beware of off-by-one errors.

3. Close the input file, and open it again, so that we can begin reading the questions and answers from the beginning. I think it is a good idea to read in all of the input data all at once rather than at scattered points during execution. So, create 2 arrays (or lists if you are working in Python) of strings: an array containing the questions, and an array containing the answers. Now, you can read

the lines of input.txt and put the strings into your arrays, as appropriate. Close the input file a second time. You won't need it anymore.

4. Now it's time to engage the user interactively with the quiz. Here you need a loop, with one iteration per question. As you ask questions one at a time, you need to keep track of how many questions the user gets right, so be sure to include a variable, e.g. numCorrect. The actual running of the quiz is pretty straightforward. Print the next question in the quiz, which is kept somewhere in your question array. Get a response from the user. Compare the user's response with the appropriate answer string from the answer array. Then, you need to determine if the user's response is "correct", i.e. equal to the answer that corresponds to the question.

   If the answer is correct, add 1 to numCorrect and print a happy message for the user. If incorrect, then you need to tell the user what the right answer was. Please refer to the example I/O given earlier in this handout. Also note the blank lines in the output for readability.

   Your program should ignore whitespace at the end of answers. The user might inadvertently hit a space before hitting enter, and we don't want to count this as incorrect. Similarly, the correct answer in the input file might also have stray whitespace. Therefore, I highly recommend that when you read answers from answers.txt, and when you read the user's interactive answers, that you trim excess whitespace. In Java, this can be accomplished by using trim() from the built-in String class.

5. Finally, the program should tell the user how many questions were answered correctly. Note that if this number is 1, you need to print "question", otherwise print "questions". If desired, you could also print the user's score as a percent.

Run your program and see how it goes!

As an additional improvement, how would you modify your program so that it shuffles the questions randomly?