Cambia Take-Home Exercise

Introduction

Thanks for applying to Cambia! This take-home exercise should give you a sense of the kind of technical skills we need to help us build world-class software. We hope you enjoy thinking about these questions and sharing your knowledge.

Try to answer in a way that helps us understand not only what you know, but also how you think. Be prepared to talk about any of these answers in a follow-up session.

Please submit answers to **all** of the exercises by checking them into a personal GitHub account and sending daniel.dinh@cambiahealth.com a link to the repository. Use GitHub in a way that demonstrates your thought process. Reach out if you have any questions.

Programming

The solution must be runnable with a Docker command (i.e., include a Dockerfile).

Imagine a CSV file called `input.csv`, which contains a single line of comma-separated strings. This single line is terminated with a new line character. Using an appropriate language, write a program that reads `input.csv`, sorts its strings into descending alphabetical order, and writes the sorted strings in comma-separated format to a new file called `output.csv`.

Here are sample contents of these two files (but your program should handle other content as well):

- input file: `Copenhagen,Stockholm,Oslo`
- output file: `Stockholm,Oslo,Copenhagen`

Gherkin

1. Write Gherkin tests for the program you wrote above. Use any Gherkin features or practices you want. Don't write step definitions (*i.e.*, the tests don't have to be executable).
2. Explain in detail why these tests might be helpful in the future.

Tools

1. In your opinion, what's helpful about version control systems? What's annoying about them?
2. What are some pros and cons of using Docker to develop, test, and deploy software?
3. How do you choose which language to use for a given task? How did you choose the language

for the programming exercise above?

Testing Methodology

1. What's the right role for QA in the software development process?
2. As a QA person, you have 2 weeks to prepare before your team starts writing software. What do you do?
3. When is it appropriate to use automated testing? When is it appropriate to use manual testing?
4. Your dev team has just modified an existing product by adding new features and refactoring the code for old features. The devs claim to have written unit tests; you're in charge of integration testing. Dedicated teams are handling performance and security testing, so you don't have to. As is always the case in the real world, you don't have time to test everything. What factors do you think about as you decide where to focus your testing efforts? How do you decide what *not* to test?

Created 2018-08-30