//

Gherkin

//

2. Explain in detail why these tests might be helpful in the future.

The Gherkin tests I wrote serve as a description of what inputs the program being created must be capable of receiving and what output the program must give out for different types of inputs.

With these tests in place, a programmer can edit the code that currently implements these requirements without fear of accidentally removing any necessary functionality. If something breaks, some of the tests will fail and inform the programmer that something went wrong. This is especially useful in terms of version control, as it keeps newly generated errors specific to the user who made them.

//

Tools

//

1. In your opinion, what's helpful about version control systems? What's annoying about them?

Version control systems are an almost necessary part of any large-scale project. They provide safety during production by allowing users to obtain older versions of code in the case where the version being worked on is lost, or it is found to be broken in some difficult to recover way. Version control also helps groups work together easily, by allowing different users to work on a single area of code in parallel through the use of branching.

What can be annoying about version control is the amount of trouble that can be caused by inexperienced users. Someone unfamiliar with a version control toolset could easily create dud branches or overwrite other's changes when attempting to push their own.

2. What are some pros and cons of using Docker to develop, test, and deploy software?

Pros:

Guarantees that all programmers are developing and testing their code in an equivalent environment through the use of a virtual machine.

Simplifies the compilation and running of programs being developed, for testing. Allows for direct deployment to the cloud.

Cons:

Containerizing software can be a complicated task for inexperienced developers, leading to slower development time.

Docker has some compatibility issues with some different languages, compilers, and IDE's. For instance, intellij uses its own system to compile its Java code, making automating the compilation process through Docker a challenge.

3. How do you choose which language to use for a given task? How did you choose the language for the programming exercise above?

I chose the language for a task based on the context in which the task would be used and the type of task that must be completed.

In the case of this task, I decided to use Java for two particular reasons:

First, Java is a compiled language. Once the program has been compiled, it will tend to run faster than a scripting language, saving time in a higher performance environment.

Second, Java supports multi-threading. In the case of file I/O and sorting algorithms, there is the potential for future optimization through multi-threading. By writing this code in Java, I leave that potential optimization open for the future.

//

Testing Methodology

//

1. What's the right role for QA in the software development process?

QA in software development is responsible for ensuring that the code being developed is produced with minimal defects. This role can be involved in every step of the coding process, working with the software developers to ensure that the design of the software is clean, that the code is written to conform to a standard, and ensuring that executing the software produces the expected and correct results through automated tests.

2. As a QA person, you have 2 weeks to prepare before your team starts writing software. What do you do?

First, I would work to ensure that the immediate expected feature set was understood, including what priority each feature has. I would summarize this feature set in a document for future reference. This includes details on the code structure for the software as well.

Then, I would implement the verbal portion of the automated testing structure, focusing on making sure the purpose of each test is clear and all features are tested as extensively as reasonable.

Finally, depending on the structure of the company, I would work to some degree on implementing a skeleton for running the testing structure. This could include references to unimplemented classes or functions depending on how detailed the code structure was before the software starts being written by the team.

3. When is it appropriate to use automated testing? When is it appropriate to use manual testing?

Automated testing is appropriate in any case where you are looking to maintain the quality of a piece of code as it is being written and updated. This is done by detecting defects or incorrect features.

Manual testing can be more valuable when you are not aiming to build quality code, such as for proof-of-concept code. It can also be helpful when you are attempting to diagnose a problem detected from an automated test, as automated testing can tend to be more focused on finding issues rather than solving them.

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?

As I would be focusing on integration testing, I would focus on my goals in the following order.

First, I would focus on immediately obvious tests. This would include documented ways that the new and refactored features are expected to work. Within these, I would further focus on features that are integral to the product, ensuring that the core features run properly.

Second, I would focus on features that have already been tested in the past, working to ensure that these features continue to work as expected in relation to the new and refactored features.

Finally, I would add any extraneous tests that appear to be missing from the test set.

In terms of what I wouldn't test, I wouldn't bother producing tests that have already been proclaimed to be written by others unless I found it necessary. If I found that the unit tests written by the developers were insufficient for some reason, I would try to resolve the issue by working with either the devs, any other experts on the product, or with any documentation that may be available, depending on what is available.