CSCE 5430, Fall 2021

Individual Assignment 4

**Due Date:** Friday, 11/5, 11:59pm

**Submission**: Canvas

# Weight: 6% of the total grade

**Assignment Description:**

1. (12 points; 3 points for each style) For each of the architectural styles described in the lecture (4 architecture styles), give an example of a real-world application whose software design might incorporate that style. Don’t describe each architectural style. Just simply provide an example for each.

Pipes-and-Filter: Image processing tools, namely Photoshop.

Client-Server: Websites. A specific example would be Canvas.

Repositories: Git services, namely GitHub.

Layering: Communication systems, such as TCP/IP.

2. (12 points) What are the advantages and disadvantages of using the same standardized language or tools across applications in your organization?

Advantages: It reduces the number of dependencies of a system, which means less overhead and the system is less likely to encounter errors due to third-party software changes. It also allows for fewer issues in getting tools or procedures to work together, as many times technologies are difficult to get to work together and require significant time and effort to make work. Additionally, this makes the system more understandable, as anyone incoming has fewer things to learn before they understand how things work. Overall this also allows for multiple systems to work together more easily than if there was no standardization.

Disadvantages: Tools and technologies are designed to do specific things, and some languages do better in certain areas than others. Standardizing reduces the number of these things across all systems, and decreases effectiveness in areas where the chosen tool, technology, or language isn’t as good at the task as a similar option. Additionally, this has the potential to reduce productivity of new hires as they might have to learn how to use things that they haven’t used before.

3. (20 points) Explain the concept of an equivalence partition, provide an example, and discuss how this concept can be used in testing.

Equivalence partitioning is where the inputs for test cases are divided into classes, so that similar inputs are grouped. This makes it easier to reduce the number of test cases by combining inputs into general test cases when possible, as the similarities are easier to find when the inputs are grouped. This makes functional testing easier to write and more dynamic, allowing for possible reuse.

An example of equivalence partitioning would be if there was general input from the user, the possible inputs could be partitioned into: string, whitespace, special character, and number.

4. (12 points) How are faults and failures related to testing and debugging?

Failures and faults are exposed during testing. Tests are specifically designed to test the limits of what a piece of code can and can’t do, and compare that to what it should and shouldn’t do. The entire purpose of testing is to expose faults and failures, and the entire purpose of debugging is to remove as many of the discovered faults and failures as possible.

5. (12 points) What are three conditions to observe software failures?

The program must be in an incorrect state, made so by being written or designed incorrectly. (Infection)

The code that will generate a failure must be executed for the software failure(s) to happen. (Reachability)

The failure must cause some sort of incorrect or undesirable behaviour from the program. (Propagation)

6. (32 points; 8 points for each sub-question) Consider the following program ***findLast***, which contains a fault.



1. Identify the fault

In the for loop, the logical evaluation is for i > 0, but the fist index is 0 and will be missed given that the loop is starting at the last index and decrementing.

1. Identify a test case that does not execute the fault. Also provide expected and actual outputs for the identified test case.

x = [1, 2, 3, 4]  
y = 3

Expected: 2  
Actual: 2

1. Identify a test case that executes the fault, but does not result in an error state. Also provide expected and actual outputs for the identified test case.

x = [1, 2, 3, 4]  
y = -3

Expected: -1  
Actual: -1

1. Identify a test case that satisfies all conditions that are required to observe software failure. Also provide expected and actual outputs for the identified test case.

x = [1, 2, 3, 4]  
y = 1

Expected: 0  
Actual: -1