Second Individual Assessment

Published | Edit | ⋮

Maximum total 100 points. There is no credit for part 0, but 0A is required in the sense that you will receive 0 overall on the assessment if part 0A is missing or incomplete. Part 0B is optional.

You are NOT permitted to post, discuss or work together on the assessment questions with other students in the class or with anyone else other than the teaching staff. Your answers can be as long or as short as you want, but a long meandering answer that incidentally mentions something relevant will receive less credit than a short high-quality answer. Submit your answers all in a single document in response to this assignment on courseworks. You can submit as many times as you want until the deadline. Submissions via email, piazza, or any other means besides courseworks will receive zero credit.

Ungraded questions: part 0A is required and part 0B is optional.

part 0A (required): You have 100 total points to allocate among the members of your team. **Assign some number of points, possibly 0, to each member of your team based on what you believe was that student's relative participation in the team project, and *explain your scores*.** For allocating points, consider level of effort and contributions for anything relevant to the team project. It is very important that you explain the scores you give to each student, including yourself: *If you do not provide meaningful explanations, then your response will be considered incomplete and you will receive 0 overall on the assessment.*

For example, in a team where all team members participated more or less equally, the scores would be: Sheldon 25, Leonard 25, Howard 25, Raj 25, and your explanation would describe who did which part(s) of the team project effort. *If you do not provide a meaningful explanation, then your response will be considered incomplete and you will receive 0 overall on the assessment.*

In a team with a disparity in contributions, however, the scores might be: Homer 40, Marge 25, Bart 20, Lisa 15, and the explanation might describe Homer as doing most of the work followed by a discussion of what the others contributed. Scores like these would most typically be provided by Homer. Marge might instead give these scores: Homer 10, Marge 50, Bart 20, Lisa 20, so we need every team members' explanations of who did what, or who did not do what they were expected to do, to figure it out. *If you do not provide a meaningful explanation, then your response will be considered incomplete and you will receive 0 overall on the assessment.*

part 0B (optional): Is there anything that you think should have been covered but wasn't? Is there anything that was covered that shouldn't have been (e.g., everyone already knows from their introductory CS courses)? Is there anything that should have been discussed more, discussed less, or presented in a different way? Is there anything else you would change about the course,

including lectures, assignments, anything else you think relevant? Suggestions for next year's offering of the course will be greatly appreciated, particularly wrt whether the individual mini-project should be assigned again or removed. Alternatives include starting the team project earlier, perhaps with three iterations instead of two, or some other kind of individual assignments for the first 3-4 weeks (suggest ideas!). Despite teaching on CVN for many years, using their microphones, the instructor has never previously encountered the microphone feedback problems that occurred in 451 CSC and she will put up a fuss to never ever be assigned that room again, so it is unnecessary to complain about that particular problem.

Graded questions: There are four graded questions, part 1-4. These questions are entirely "on paper"; do not modify your team's code to fit the code to these questions, instead fit your answers to your existing code where applicable.

Part 1 (20 points): You are a member of a team developing the XXX software (assume some arbitrary software project, not your team project for this course). Your team arrives at the scheduled time to give your demo. Your software is hosted on YYYCloudPlatform and cannot run on localhost. You begin to startup your software, but the cloud host displays an odd error message you've never seen before and nothing further happens. You try again, same thing. Everything worked fine last night when your team practiced the demo. Finally, one of your team members admits that they made small changes to a few files after your practice session, committed to your shared repository, and uploaded the new build to your cloud host - overwriting the previous build you practiced with yesterday. This team member has often changed your software right before an important demo, but their previous changes always worked.

Take one of these two positions, A or B:

A. This scenario is, unfortunately, likely to happen and explain what your team would do to fix the problem quickly to proceed with the demo.

B. This scenario is, thankfully, unlikely to happen and explain what your team set up to prevent it.

A trivial answer like "We'd reschedule the demo" or "The overeager team member does not know the cloud host password" will receive 0.

Part 2 (40 points – 30 points for 2a and 10 points for 2b): 2a and 2b can refer to the same or different parts of the code.

Part 2a (30 points): Pick a non-trivial class or module from your team project, consisting of at least three non-trivial methods/functions, and state which class/module and methods/functions you're considering. If your program does not have any components with a least three non-trivial methods, pick three non-trivial methods from different components and clarify exactly which ones. By non-trivial, this question means methods including conditional branches and/or loops.

Describe the input equivalence partitions for the three methods and how, ideally, you would test each of those partitions. Recall that functions with multiple parameters usually have separate sets

of equivalence partitions for each parameter. Make sure to include invalid as well as valid partitions. Do not modify your team's code.

(If your team project does not include at least three non-trivial methods, contact the instructor asap. She will check your team repository and then provide an alternative question if warranted.)

Part 2b (10 points): Did your team actually employ input equivalence partitioning when designing your unit tests and/or system tests? Do not modify your team's code.

If so, please tell us which specific tests in your codebase address which specific partitions (you only need to cover three units and/or API entry points for this question).

If not, describe what approach your team did use to invent test case inputs (again consider three units and/or API entry points).

Part 3 (20 points): Did your team use any design patterns in writing your code? This might include design patterns listed in any design patterns catalog whether or not they were discussed in class.

If your team used at least two design patterns, pick any two distinct design patterns, point to the specific code that reflects each of them, and describe in each case how it implements that design pattern.

If you used only one design pattern, point to the specific code that reflects it and describe how it implements that design pattern. In addition, pick a second design pattern and describe how your team could have used it to improve the quality of your code.

If your team did not use any design patterns, pick two distinct design patterns and describe how your team could have used each of them to improve the quality of your code.

Do not consider MVC as a design pattern (you will receive 0 for this question if you do) and do not modify your team's code.

Part 4 (20 points): Imagine that your team project was required to develop a *library* instead of a *service*, with the intent that the library could be used by other developers as part of their own application or service.

What are the most significant changes that would be needed in the software architecture, API and persistent storage of your system and why?

Ignore whether your system's functionality might not make sense for a library, this question is concerned with the structure of your software, not what it does. Do not modify your team's code.

This is the end of the assessment.

Points	100
Submitting	a file upload

File Types pdf, doc, docx, and txt

Due	For	Available from	Until
Dec 19, 2021	Everyone	Dec 16, 2021 at 12:01am	Dec 20, 2021 at 11:59pm

Second Assessment

You've already rated students with this rubric. Any major changes could affect their assessment results.

Criteria	Ratings		Pts
1. Option A: Version control and reverting. / Option B: Continuous Integration [Option A: scenario is, unfortunately, likely to happen] 6 points for discussing the concept that version control supports reverting to a previous version of the source code as it was at some particular point in time (or a particular tag). [Option B: This scenario is, thankfully, unlikely to happen] 6 points for discussing the concept that CI can be configured to run automatically on commit (or when commit is attempted).	**6 pts** **Full Marks**	**0 pts** **No Marks**	6 pts
1. Option A: Which point in time to revert to / Option B: CI rejects commit [Option A: scenario is, unfortunately, likely to happen] 7 points for discussing how they figure out which point in time (or tag) they want to revert to, presumably what they had practiced with the night before, This is not necessarily the most immediate previous commit or tag, since the team member might have made a series of commits and possibly tagged them. [Option B: This scenario is, thankfully, unlikely to happen] 7 points for discussing the concept that CI supports rejecting a commit if any analyses, tests, etc. fail (or alternatively it accepts the commit but blocks deployment, which	**7 pts** **Full Marks**	**0 pts** **No Marks**	7 pts

Criteria	Ratings				Pts
would work if the CI tool also automates deployment).					
1. Option A: Rebuild and redeploy / Option B: Discussing unit test, analysis tools [Option A: scenario is, unfortunately, likely to happen] 7 points for discussing that they have to rebuild and redeploy the software from that previous version, i.e., the version repository does not store builds and the executable does not magically appear on their cloud host (which does not store backups). [Option B: This scenario is, thankfully, unlikely to happen] 7 points discussing what CI does wrt running unit tests and analysis tools, which have to be written, configured, etc. in advance.	**7 pts** **Full Marks**		**0 pts** **No Marks**		7 pts
2a. Equivalence Partition - Method 1	**10 pts** **Full Marks**	**5 pts** **Incorrect Invalid partition** 5 points for covering all main invalid equivalence partitions you can think of for that method. There might be only one or there might be several. If you cannot think of any invalid partitions, ask me to look at it.	**5 pts** **Incorrect Valid partition** 5 points for covering the main valid equivalence partitions you can think of for each parameter of that method. There might be only one or there might be several.	**0 pts** **No Marks**	10 pts
2a. Equivalence Partition - Method 2	**10 pts** **Full Marks**	**5 pts** **Incorrect Invalid partition** 5 points for covering all main invalid equivalence partitions you can think of for that	**5 pts** **Incorrect Valid partition** 5 points for covering the main valid equivalence partitions you can	**0 pts** **No Marks**	10 pts

Criteria	Ratings				Pts
		method. There might be only one or there might be several. If you cannot think of any invalid partitions, ask me to look at it.	think of for each parameter of that method. There might be only one or there might be several.		
2a. Equivalence Partition - Method 3	**10 pts** **Full Marks**	**5 pts** **Incorrect Invalid partition** 5 points for covering all main invalid equivalence partitions you can think of for that method. There might be only one or there might be several. If you cannot think of any invalid partitions, ask me to look at it.	**5 pts** **Incorrect Valid partition** 5 points for covering the main valid equivalence partitions you can think of for each parameter of that method. There might be only one or there might be several.	**0 pts** **No Marks**	10 pts
2b. Equivalence Partition - Project [Yes] +10 point if code and tests matches equivalence partitions. These might or might not be the same as in 2a, either way is ok. [No] +10 point if code and tests match stated 'approach'. One good approach is choosing inputs to force branches, improving coverage. Choosing inputs at random is not a good approach, but still give full credit if it indeed looks like that's what the code did.	**10 pts** **Full Marks**		**0 pts** **No Marks**		10 pts
3. Design pattern 1 5 points for explaining the gist of the design pattern. 5 points for describing some plausible way the team did use the design pattern or could have used the design pattern if they didn't already.	**10 pts** **Full Marks**	**5 pts** **Missing 1 of the 2 items**	**0 pts** **No Marks**		10 pts

Criteria	Ratings			Pts
3. Design pattern 2 5 points for explaining the gist of the design pattern. 5 points for describing some plausible way the team did use the design pattern or could have used the design pattern if they didn't already.	**10 pts** **Full Marks**	**5 pts** **Missing 1 of the 2 items**	**0 pts** **No Marks**	10 pts
4. Architecture changes The gist is a service runs as a separate program whereas a library runs as part of someone else's program. The service program runs on a platform the student's team configures and generally controls, whereas a library has to run on any platform configured to execute the program in which the library is included. The answer should be tailored to the student's team project.	**8 pts** **Full Marks**		**0 pts** **No Marks**	8 pts
4. API changes A service receives inputs over the network using a protocol like REST or RPC. In contrast, a library receives inputs as parameters to function calls. If the student's team built their service using some framework that hardwires how inputs are handled, it probably has to be removed. The answer should be tailored to the student's team project.	**8 pts** **Full Marks**		**0 pts** **No Marks**	8 pts
4. Persistent storage changes There are real-world libraries that connect to a remote database shared by all instances of the library, there are libraries that insist on being configured with a local instance of some specific database, there are libraries with multiple interfaces to work with	**4 pts** **Full Marks**		**0 pts** **No Marks**	4 pts

Criteria	Ratings		Pts
whatever data store is available locally, and so on.			
			Total Points: 100