SE2030 Half-Exam 2 Name:

Between the time you start and finish the exam, you may not contact any human other than Dr. Yoder (your instructor for this course). This includes in person, by phone, through the internet, or any other media. Violation of this requirement is a serious violation of this course's integrity policy (https://faculty-web.msoe.edu/yoder/se2030/#integrity). Also, between the time you start the exam and the "due date" for the exam, you may not discuss this exam with any other person. Again, even "minor" violations of this requirement could be considered serious violation of this course's integrity policy.

This is an open-notes, open-internet, closed-human exam. (The previous paragraph describes what closed-human means.)

Topics for this exam:

The full list of outcomes for the course can be found on our course webpage: https://faculty-web.msoe.edu/yoder/se2030/Outcomes

The examples given here are drawn from the outcomes which were not included on Blackboard Quiz 1, Blackboard Quiz 2, Half Exam 1 and Half Exam 2. Please review those quizzes and exams in addition to looking over this practice exam to get full coverage in your study for the final. This is a comprehensive final, and I will put the most points on the core topics that we have studied and discussed the most.

- Writing unit tests
  - Provide an example of when the assertThrows test would be useful
- Version Control
  - Describe the advantages of a VCS system over a shared hard-drive / usb drive / network drive or Google Docs, Office 365, etc.
  - Describe the lock-modify-unlock procedure
  - Describe the situations in which the lock-modify-unlock procedure may still be useful
- Git
  - Explain how distributed version control impacts the standard git workflow.
  - Describe the purpose of the stage in making a commit
  - Describe the purpose of a commit — a repository history entry
  - Use a text editor and the basic Git commands to resolve a merge conflict
  - Review your work with the git status and git diff commands before staging and committing your code
  - Use the xkcd approach to resolving a Git mess. (The Git equivalent of turn it off and turn it back on :-))
- Advanced Git Topics (branching, merge requests, Git Flow)
  - Explain the motivation for keeping the master branch separate from the dev branch

  - Describe the advantages and disadvantages of creating a feature branch
  - Describe how a Git tag can be used within Git Flow
  - Describe the advantages and disadvantages of a hotfix or a release branch. (These are also illustrated within Git Flow)
- Defect Tracking
  - Explain why the term "defect" leads to better root-cause analysis than the term "bug"
  - Explain the advantage of detecting defects early in the development process
  - Explain how a defect would flow between the New, Valid, Disputed, WIP, Fixed, and Closed states.
  - Explain why, when tracking the rate that defects are fixed, we look at the rate of defects going from Fixed to Closed rather than from WIP to Fixed.
  - Describe how the known to unknown defect ratio may be estimated in practice
  - Describe how the known to unknown defect ratio may change as QA (testers) and coders are added to a project.
  - Define "Technical Debt"
  - Expound upon the similarities between technical and financial debt and the impact this has on project planning
  - (optional) Describe how technical debt can be addressed while continuing to develop other forms of value for the customer and the company
  - Give examples of avoidable and unavoidable technical debt
  - Give examples of strategic and irresponsible technical debt

1. (10 points) ***Describe*** how the lock-modify-unlock procedure allows users to work on the same repository without finding merge conflicts.

2. (10 points) ***Give an example*** of a difficulty that can be encountered with the lock-modify-unlock paradigm

3. (20 points) In the copy-modify-merge framework, merge requests are often automatic, but sometimes merge conflicts occur. ***Give three examples*** of how the team can all work on the project at the same time yet avoid getting into as many merge conflicts.

4. (30 points) Draw a sequence diagram. See feedback on Half Exam 2!!!
5. (30 points) Draw a class diagram. See feedback on Half Exam 1!!!

6. (20 points) Draw a use case diagram. This has not been tested this quarter. See slides on use case diagrams and note that they are different than written use cases.
7. (20 points) In Git Flow, a release branch is separated from the dev branch and eventually merged into the master branch.
    a. Describe how the coding effort on the release branch will differ from the dev branch.
    b. Choose one:
        i. The dev branch is continuously merged into the release branch
        ii. The release branch is continuously merged into the dev branch
        iii. Explain how the answer you selected contributes to a better quality release.
8. (20 points)
    a. Describe the disadvantages of having a feature branch run on for several weeks
    b. Describe a strategy your team could use to make the merge simpler while still allowing the feature branch to run for weeks without intermediate merges.
9. (20 points)
    a. ***Circle one***. Assuming all other aspects of a project are fixed, increasing the number of QA testers will cause the known/total bug ratio to go
        i. Down
        ii. Up
        iii. Stay about the same
    b. ***Explain*** your answer.
10. (20 points)
    a. ***Circle one***. The hardest error to correct will generally be an error introduced during the
        i. Requirements
        ii. Design
        iii. Implementation
        iv. Testing
        v. Maintenance

        phase of a project.
    b. ***Explain*** your answer.
11. (20 points) Give an example of
    a. Avoidable technical debt
    b. Unavoidable technical debt