CS409 - Machine Problem (MP1)
JUnit and Test Driven Development
Total: 20 points
Deadline: October 10, 11.59pm

The objective of this machine problem is to:
1. Provide students with opportunity to learn how to commit and push your changes to a GitHub repository
2. Provide students with opportunity to learn how to write JUnit tests
3. Provide students with opportunity to learn about creating an Android app

In this machine problem, you will learn how to create a simple calculator app by watching a tutorial. While watching the tutorial, you will use GitHub for getting the source code of the app mentioned in tutorial. Then, you will learn about Test Driven Development (TDD) by writing test first before implementing the functionality. If you have questions regarding this MP, read the post at
https://github.com/orgs/cs409-software-testing2020/teams/allstudents

Note that this MP requires writing JUnit tests, not GUI tests. You may need to refactor (Using "Extract method") to be able to test each method
What to submit?
***Include your student id in your README.md.**
***Do not submit *.apk**
***Add .gitignore to exclude some unwanted files (ignore apk and other class files)**
***Submit your source code and test cases in each commit as mentioned in the instruction.**
***Please follow the step-by-step instruction for each commit**
***Read the following link for the common questions and answers for the MP1 before asking questions:**
https://github.com/orgs/cs409-software-testing2020/teams/allstudents/discussions/5

i. Watch the tutorial on how to create a simple calculator app at the android-app-tutorial.zip in Sakai->assignments folder.
ii. Copy and paste this link: https://classroom.github.com/a/js3JCXJl to a browser to clone your own Github repository. This repository contains the starter code in the tutorial.
iii. Add a JUnit test for checking numbers that involves decimal points (i.e., a number with ".", such as 1.2). This test serves as a specification for the goal of adding a new button "." for adding the functionality to support entering decimal points. Add a comment /* Initial test for TDD */ to mark this test. **Commit this test to GitHub with the commit message "Add TDD test1".** *(2 points)*

iv. Add little code to make the test pass in Step iii. **Commit this code to GitHub repository with the commit message "Initial code for passing TDD test".** *(2 points)*

v. Refactor the code for adding the functionality to support entering decimal points. In this step, you should remove any code duplication. *(2 points)*

vi. Add one JUnit test for each arithmetic operation ("+","-","*","/") and **commit this test**. *(4 points)*

vii. Implement the logic for the decimal point button "." and **commit this test** (4 points)

viii. Add one JUnit test for checking exceptional behavior (An exception is expected) **and commit this test.** *(4 points)*

ix. Add one JUnit test for checking invalid input **and commit this test.** *(4 points)*

x. Compile your modified app and attached a screenshot of your app in the README.md *(2 points)*