CS409 Project: Progress Report *(Total: 20 points + 5 bonus points)*
Deadline: December 4 (Friday), 11.59pm

You have learned different ways of modeling software, including input space, graphs and logic. The goal of this progress report (Invitation link: https://classroom.github.com/g/B4ClKVIX) is for you to summarize what you have learned and your findings.

List the names and student ids of the members in your group. State clearly which student is in charge of which app.

All answers should be written in README.md.

1. In "mp2-part2", you have chosen a method for performing input space modeling. This question encourages you to think about constructing JUnit test cases at the GUI level.

a) Add the selected method in the README.md by quoting the source code.

b) Construct test cases that fulfilled Each Choice Coverage at the GUI level (*If your JUnit test cases couldn't be constructed at the GUI level, explain the reason. For example, if there is no button/option to enter a negative value at the GUI level, then this JUnit test case is infeasible at the GUI level.*). *These test cases can be either hand-written (e.g., "Edit Text field with username=ccs409"->"Edit Text field with password=12345"->"Click button Login".) or record-and-replay using MonkeyRunner/Espresso.* *(4 points)*

2. Draw the event flow graph (a simplified definition of event-flow graph is given as below) of your selected app. **Event-flow Graphs (EFG):** Each node in an EFG captures a specific user interaction (e.g. touching a button on smartphone screen), whereas an edge in the EFG captures a possible transition between two user interactions. *If your EFG is too complicated (with more than 100 nodes), draw a partial EFG that captures the important events from the main screens.* *(4 points)*

a) Construct 2 test cases by traversing (clicking-through) your EFG. *These test cases can be either hand-written (e.g., "Edit Text field with username=ccs409"->"Edit Text field with password=12345"->"Click button Login".) or record-and-replay using MonkeyRunner/Espresso. (4 points)*

3. In *"logic-sourcecode-lab",* you have selected a method with at least two clauses in your selected app where you write JUnit tests for fulfilling various logic coverage criteria.

a) Add the selected method in the README.md by quoting the source code.

b) Construct 2 test cases at the GUI level to fulfill **predicate coverage** criteria. *These test cases can be either hand-written (e.g., "Edit Text field with username=ccs409"->"Edit Text field with password=12345"->"Click button Login".) or record-and-replay using MonkeyRunner/Espresso. (4 points)*

4. Compare the three methods *(input domain modeling, graph coverage, and logic coverage)* of constructing test cases by answering the following questions. *You should discuss this question with your teammate after completing question 1-3.*

a) Which method is the most useful in finding new bugs? Why? (2 points)

b) Comparing input domain modeling and logic coverage, which method is the most effective in constructing test cases at the GUI level? (2 points)

You can do either of the following to get bonus points (5 points):

a) If you find a bug, post the bug that you find in GitHub by posting in your team discussion. If all your team members checks that your bug reports fulfilled the bug reports requirement, post it at: https://github.com/orgs/cs409-software-testing2020/teams/allstudents/discussions/

b) If you read a GitHub issue in the GitHub Discussion (Team or All-Students discussion) within your group and could use the information in the GitHub issue for finding a "similar" bug in your selected app, comment in GitHub by posting at: https://github.com/orgs/cs409-software-testing2020/teams/allstudents/discussions/2*.You need to add a bug reports (check the bug report requirement) and add a comment on "What information is useful for sharing this bug?"*