Instructions for Second Coding Assignment 1DL610 HT24

1 General Instructions

Please upload your work to studium by 1700 on 20 December 2024. This assignment is worth 10 points, points for substasks are as indicated. In case there are $i < 4$ people in a group, you may choose to do any subset of size i of the tasks. Book an appointment with the assigned TA before the 3 January 2025. **Please note it will be difficult to accommodate extensions, since the TAs may not be available later.**

2 Tasks

The first two tasks use the code from assignment 1. The third and fourth tasks use a new file included with this assignment: air_traffic_control.py **Create a separate document per task which explains your work, addressing the questions posed under that task.**

2.1 Tasks 1 and 2

Task 1
Implementation 1: Strengthening the user profile. (3 points)
Expand the user file with credit card details for an arbitrary number of credit cards. Each card has a card number, date of expiry, name on card, CVV for each card. Add a module that allows the user to change these details in the file. Expand the user registration to allow entry of these additional details.

Implementation 2: Strengthening the payment system. (2 points)
At the point of payment, ask the user whether he wants to use his wallet or a card. If "card" is selected, display the list of his cards and ask which card the user wants to use.

Regression testing. (5 points)

- Create a regression test suite which consists of 5 tests for each function that is affected by your implementation, selected from the set of unit tests you have already created for assignment 1. In addition, create 3 more tests per new functionality for each of the two new implementations i.e. 6 additional tests in total.

- Run the test suite before implementations and enable recording of the results in a file, say logfile1.
- After doing the implementations, run the test suite again and record the results in logfile 2.
- Do a diff between logfile1 and logfile2.

Why did you choose the 5 tests you did? What do you see from the diff operation prior to and post implementations?

Task 2
Creating the Control Flow Graph. (2 points)
You will need to draw the CFG of the program from assigment 1 to come up with tests. You could do it by hand, or use some existing software (such as pycfg, as explained here: `https://medium.com/@vinoothna.kinnera/creating-control-flow-graphs-using-pycfg-ba84311ca59`) or staticfg (`https://pypi.org/project/staticfg/`). Submit the CFG you have drawn.

Identifying Prime Paths. (3 points)
Identify 10 prime paths in the CFG you created.

Creating a test suite. (5 points)
Create a test suite with 10 test cases that covers all of the 10 prime paths that you identified, the test path is allowed to tour the prime path with sidetrips and detours. Identify in each case whether the test path has sidetrips, detours, both or neither.

Can you demonstrate the path taken by the tests and explain why they cover the prime path they are supposed to be covering?

2.2 Tasks 3 and 4

Use the given file air_traffic_control.py
Note that the code contains some derived predicates and the conditions in the code use these. If you consider the predicate $p =$ `runway_available and safe_speed and not emergency and safe_weather and acceptable_traffic` note that `runway_available` is itself comprised of two clauses `runway_clear, alternate_runway_available`. You should think of p as using these two clauses, `runway_available` is not a clause.

Task 3
Predicate coverage. (3 points)

Create a test suite that achieves predicate coverage. The set of predicates to be considered are those in the if conditions in the decision making process.

Clause coverage. (7 points)
Create a test suite that achieves clause coverage for a selection of 7 clauses of your choice. That is, for each clause c ensure there is a test that sets it to true and one that sets it to false. Why did you choose these 7 clauses? Does your

test suite for clause coverage subsume predicate coverage?
Task 4
Active clauses. (3 points)
Identify 3 active clauses i.e. a clause c and a valuation f of the clauses in that particular predicate such that if the truth value of c is changed while retaining the truth value of the other clauses, then the truth value of the predicate also changes.
Test suite for CACC. (3 points)
Create a test suite that achieves CACC restricted to the 3 active clauses you identified. That is, for each identified clause c, there is a test t_1 which sets c to true and a test t_2 which sets it to false, and further the value of the predicate is different between the two tests.
Test suite for inactive clause coverage. (4 points)
Identify 2 inactive clauses i.e. a clause c and a valuation f of the clauses in for a particular predicate such that if the truth value of c is changed while retaining the truth value of the other clauses, then the truth value of the predicate **does not** change. Create a test suite that achieves Restricted Inactive Clause Coverage for the two inactive clauses that you have identified.

Why do your two test suites differ? Can you explain how your test suites achieve the targeted coverage?