ASSIGNMENT 2 [40%]
White box testing and code analysis

Overview
For this assignment, your task is to design and document appropriate tests for a software system using white box techniques, build a CI/CD pipeline to run your tests, and report on the code quality and test coverage. In doing this **you must only use concepts that have been covered in FIT2107**.

This assignment is an **individual, open book** task. Every student must complete and submit their own work. The use of AI in any way is not permitted.

Submissions will be marked out of 40, and will form 40% of your final grade in FIT2107. A late penalty of 5% per day will be applied, and after 7 days a mark of 0 will be given and no feedback will be provided on the submission.

This assignment covers FIT2107 learning outcomes 1, 2, and 3.

Deliverables
You must submit the following evidence using the Assignment 2 submission page on Moodle:

- **Your tests**
  Compress your "tests" folder from "bat" to a zip named "tests.zip".
- **Task 6 and Task 9 documents**
  Convert both documents to PDF for submission, and do not rename them (i.e., you should submit files named "task_6.pdf" and "task_9.pdf").
- **Task 3 control flow graph**
  Submit as "task_3.png"

Your grade will also be based on the commit history in your unit repository on gitlab.

SUBMISSION DUE: Friday Week 12, 11:55PM

Tasks

For this assignment you are continuing to test the *Borrowing Administration Terminal* (BAT) used by *Anything Anytime Library* (AAL). However, now you will have access to some of the BAT code. This will enable you to run additional tests on the system using white box techniques.

The code you have been provided with is a redacted version of BAT. Each part of the code has been commented to describe its purpose and intended logic. In addition to the in-code documentation, you should note:

- It is assumed that a patron will never attempt to take out a loan for an item they are already borrowing (e.g., borrow two copies of the same book).
- It is assumed that there are no patrons with the same name and age.
- It is assumed that there are no logic errors in the JSON data provided to BAT (e.g., duplicate IDs, loans which aren't reflected in the catalogue). If there are any syntax errors in the data then BAT will not open.
- Changes to data are not saved until the "Quit" menu option is selected.
- All functionality to do with late fees has been removed, except the calculation of discounts for the purpose of determining if a patron is allowed to borrow an item or is not allowed due to fees owed.
- Ability to update training records has been removed.
- All analytics code (e.g., for generating overdue loans reports) has been removed.
- All user and catalogue data is fabricated.

Task 1: File Setup (Marked as part of development history modifier)

You should have the knowledge to complete this task after Week 1.

Download the assignment template (template.zip) from Moodle. Unzip the folder, and copy all of the files into the "Assignment 2" folder in your unit repository. Add, commit, and push the files.

Running BAT:

Assuming you have set up your files correctly, you can run BAT by:

1. Open your unit repository in VS Code.
2. Open VS Code's integrated terminal.
3. Use **cd** to navigate to the "Assignment 2/bat" folder.
4. Run the terminal command "python run.py".

If you ever get stuck or want to exit BAT quickly, you can use the shortcut **control+c** in the terminal to terminate the currently running process.

Task 2: MC/DC (4 marks)

You should have the knowledge to complete this task after Week 7.

In the file "test_task_2.py" in the "tests" folder in "bat", write tests for the method "can_borrow_carpentry_tool" from the "business_logic.py" source file. Write the minimum number of tests needed to reach 100% MC/DC for the condition on line 126. Do not write any extra tests.

Document your tests in comments as shown in the week 7 answer guide for MC/DC. Your documentation must:

- List the possible tests and their outcomes, numbering each test.
- List the optimal test sets.
- Clearly identify which optimal set you have chosen to implement.
- Clearly identify which test number is being tested by each test method.

Task 3: Path Coverage (5 marks)

You should have the knowledge to complete this task after Week 7.

Part A:

Draw a control flow graph for the method "can_use_makerspace" from the "business_logic.py" source file. In your graph, show "else" explicitly as a node. Save your graph in the "Assignment 2" folder in your repository as "task_3.png".

Part B:

In the file "test_task_3.py" in the "tests" folder in "bat", write tests for the method "can_use_makerspace" (from the "business_logic.py" source file). Write the minimum number of tests needed to reach 100% path coverage. Do not write any extra tests.

Document your tests in comments as shown in the week 7 answer guide for path coverage. Your documentation must:

- List the feasible paths, number each.
- Clearly identify which path is being tested by each test method.

Task 4: Mocking (5 marks)

You should have the knowledge to complete this task after Week 8.

In the file "test_task_4.py" in the "tests" folder in "bat", write tests for the method "_main_menu" from the "bat_ui" source file. Use the method "get_current_screen" to verify that the UI has moved to the correct screen. Do not write any extra tests.

Make sure you test that:

- All valid inputs cause the UI to move to the correct screen.
- The user is repeatedly asked for input until a valid input is given.

Task 5: Coverage (10 marks)

You should have the knowledge to complete this task after Week 8.

In the "tests" folder in "bat", write tests for any of the code in BAT you like until you reach:

- At least 90% statement coverage.
- At least 80% branch coverage.

All tests written for this assignment (i.e., not just for Task 5) count towards coverage.

Add as many test files to the "tests" folder as you need, but do not add tests to the "test_task_X.py" files, and do not rename any of the "test_task_X.py" files. Part of your grade for this task will be based on your ability to write appropriate tests, and organise them into appropriately named test files. Make sure you follow all guidelines given in this unit.

Note: to get just branch coverage, run the terminal command "coverage json" after a "coverage run" command. This will generate a file ("coverage.json"). In that file, under "totals" there'll be "covered_branches" and "missing_branches".

Task 6: Find the Bugs (9 marks)

You should have the knowledge to complete this task after Week 8.

Part A:

Using any technique or combination of techniques you like, find 3 bugs in BAT.

Part B:

In "task_6.docx" in the assignment template, write a bug report for each of the three bugs you found.

Task 7: CI (2 marks)

You should have the knowledge to complete this task after Week 9.

Configure gitlab to automatically run all of the tests you have written for BAT. Ensure that only the BAT tests run, and the output is verbose. Gitlab should show that all your tests were run, and all your tests pass.

Task 8: Static Analysis (2 marks)

You should have the knowledge to complete this task after Week 9.

Update your gitlab configuration to also automatically run the "pylint" and "pycodestyle" static analysis tools on the BAT source code and tests. Gitlab should show that all your tests were run and pass, but the code does not have to pass linting.

Task 9: Software Metrics (3 marks)

You should have the knowledge to complete this task after Week 10.

As mentioned, you have been given a redacted version of the BAT code. The real version of BAT has 9842 lines of code.

In "task_9.docx" in the assignment template, explain (including your working) how many defects you would expect the real version of BAT to contain. Your answer should not be longer than one page.

Assessment Criteria

This assignment will be marked out of 40, and will form 40% of your final grade in FIT2107. A late penalty of 5% per day will be applied, and after 7 days a mark of 0 will be given and no feedback will be provided on the submission.

- **Development history**
  After your submission is marked, a modifier will be applied to your score based on your development history. The lowest possible modifier is 0.5, and the highest possible modifier is 1.0 (i.e., no grade reduction). Your final grade will be your original grade multiplied by this modifier. To get a modifier of 1.0 you need to:
  - Have all the files from the assignment template in the "Assignment 2" folder in your unit repository, and not in a sub-folder.
  - Make at least 11 commits total.
  - Make at least 2 commits of each file in the template.
  - Use meaningful and concise commit messages.
- **Appropriate use of unit concepts**
  The marker will verify that you have used only concepts covered in FIT2107.
- **Correctness**
  The marker will verify the correctness of your answers.

- **Clarity**
  The marker will verify whether your answers use clear, specific, and appropriate examples. This is particularly important when you are writing justifications.
- **Consistency**
  The marker will verify whether related answers are consistent with each other.
- **Good coding practice**
  The marker will verify whether you have followed good coding practice for writing tests in python, as demonstrated in this unit.
- **Test success**
  The marker will verify whether your tests pass, and can be run repeatedly with the same results.