EECS4315 (Section Z) Winter 2025
Guide to Programming Test 1

When: Thursday, February 13 (during your enrolled lab session)
Where: LAS 1006
Duration: 50 minutes (9:00 AM to 9:50 AM)

Chen-Wei Wang

Last Updated: February 2

1 Policies

- This programming test is in-person and **strictly** individual: plagiarism check may be performed and suspicious submissions will be reported to Lassonde for **a breach of academic honesty**.
- This programming test will account for **10%** of your course grade.
- This test is **primarily** a programming test: it will assess if you can write
    - **valid** `PlusCal` algorithms and assertions **free of** syntax, type, and logical errors; and
    - precise & concise answers to questions requiring the inspection of some given `PlusCal` algorithm.
- **Structure of the Test**:
    - At 9:00 on the test day, all lab machines will be rebooted to the "lab-test mode" (where there is **no** network connection and you are expected to use the **TLA+ toolbox only**).
    - During the test, you will be expected to launch the TLA+ toolbox on a designated workspace.

      You may be asked to:

      1. Create modules with the instructed names. For each module that you create:
          - ∗ Implement an algorithm for the given problem, with the **required** input and output variables.

            Caveat. It is absolutely critical for you to use the **exact names** of input and output variables as required by the test instructions (otherwise, grading assertions using these names will fail to work on your submitted modules).
          - ∗ Specify assertion(s) that correctly (and **completely**) formulate the described preconditions and/or postconditions of the algorithm.
          - ∗ Auto-translate the written `PlusCal` algorithm and assertions into TLA+ syntax, and use the TLC checker to verify the algorithm against the assertions you write.

            Caveat. It is absolutely critical for you to be sure that all loops you write **terminate**; otherwise, a non-terminating loop may cause the TLC checker to hang indefinitely, and you will be responsible for any time wasted to kill the process and to re-start the tool (or to even re-start the lab machine).
      2. Import a given module. Some example tasks are:
          - ∗ Identify and/or explain some logical errors (e.g., failed checks on some invariant).

    - Extend the module (with PlusCal code and/or Boolean constraints) to perform certain tasks (e.g., non-deterministic event selection, system variant checks).

  - You are **solely responsible** for:
    - **leaving enough time ($\approx$ 3 minutes) to export the completed .tla module files and upload/submit them to WebSubmit**; and
    - **submitting the right module files for grading**.
  - During the test, instructions will be given to remind you of how to setup the **state graph** generation tool (whereas there will be **no** help on how to use the tool and how to access/interpret the generated graph)

- **Submission for Grading**:
  - Like your assignments, submission (of module `.tla` files) for this programming test must be through the WebSubmit link (which will be provided during the test).
  - It is your sole responsibility for making sure that the correct version of each module files is submitted. **After** clicking on the `submit` button on WebSubmit, you should **re-download** the module files and make sure that they are the right ones to be graded. **No** excuses or submissions will be accepted after your attempt times out.

- **Grading Criteria**

  In each module file that you are required to submit:

  - If your submitted file, loaded in the TLA+ toolbox and auto-translated to the TLA+ model, contains any errors (due to e.g,. syntax errors, type errors):

    TAs will attempt to fix the errors for you, if they can be fixed quickly. If succeeded, there will be a ***20% penalty*** on the allocated marks for that module.
  - If the input and output variables you use are **not** exactly as instructed:
  - TAs will attempt to fix this for you. If succeeded, then there will be a ***10% penalty*** on the allocated marks for that module.
  - To assess your written answers (as comments), TAs will grade them manually.
  - To assess the correctness of your `PlusCal` algorithm, we will check it against (automatically using the TLC checker) our assertions (which are consistent and complete with respect to the problem descriptions give to you).
  - To assess your specified assertions, we will:
    - grade them manually for its completness (i.e., whether or not there are missing cases) and correctness (i.e., whether or not the logic is correct); **and/or**
    - grade them automatically by:
      - replacing your algorithm with an **incorrect** algorithm
        (and see if your assertions would **fail** as expected); and
      - replacing your algorithm with a **correct** algorithm
        (and see if your assertions would **pass** as expected).

2 Format

The format of this programming test will be mostly **identical** to that of your **Lab2**: given informal problem descriptions (on the required inputs, outputs, and input-output relations), implement `PlusCal` algorithms and specify the appropriate assertions (which formulate the algorithm's preconditions and/or postconditions).

- As a reminder of the basic syntax, the following document will be made available to you during the test:

  `https://d3s.mff.cuni.cz/f/teaching/nswi101/old/pluscal.pdf`

  You're advised to go over the above document prior to the test so that you can easily find what you need during the test.

- You will have access to the tool for generating a **state graph**.

- You will be expected to create models (of modules) and verify their correctness (via the TLC checker). To create models (by instantiating constants), see Lab1.

3 Coverage for the Test

- Lab1

- Lab2

- You do not need to review lecture materials.

4 Mockup Test & Practice Questions

- A **mockup test** is scheduled during the lab session (9am to 9:50am) on Thursday, February 6.

- During this mockup test, some practice questions will be given to help you familiarize yourself with the test format and flow.

- Solutions to these practice questions will be made available by the end of Friday, February 7.

- This mockup test will **not** be graded, but you will get a chance to practice submitting it.

- It is important to note that these questions are meant for familiarizing yourself with the **format** and **workflow** of the test, and they represent **only** as an example:

  - You are expected to study **all** materials as listed in Section 3.
  - The actual test may cover aspects not covered by the practice questions.
  - Some questions in the actual test may be harder than the practice questions.

5 Simulating the Programming Test (after the Mockup Test)

It is highly recommended that you simulate taking the programming test by following these steps:

Preparation

- Login into a machine under remotelabs (using your EECS account): `https://remotelab.eecs.yorku.ca/`. Choose a machine under the `ea` category.
- Launch the Firefox web browser (under Activities) and login into the Section Z eClass site.
- Download and open the `PracticeTest1.pdf` file from eClass onto the Desktop.

Start the Test

- Start a timer (say for 50 minutes).
- Launch TLA+ toolbox (under Activities)
- Tackle the test by: creating the modules as instructed, implementing algorithms, and specifying assertions.
- **Before you submit, you should make sure that there is no error in any of the module files.**

Submission

- **It is a recommended practice that you submit intermediate versions of your developed modlues (e.g., every 15 to 20 minutes).**
- Upload all required `.tla` module files to the WebSubmit link for grading:

  `https://webapp.eecs.yorku.ca/submit/?acadyear=2024-25&term=W&course=4315&assignment=PT1`

  The above link will be made public when the practice questions are released.