Testing your lab code

Quick Links: Lecture material (https://q.utoronto.ca/courses/419441/pages/lecture-material) Lab assignments (https://q.utoronto.ca/courses/419441/pages/lab-assignments) Piazza Discussion (https://piazza.com/utoronto.ca/winter2026/ece353/home)

The ECE353 AutoTester

We have provided you a program called `ece353-tester` that allows you to 1) test your code with various test cases, and 2) allows simulating the way we will be marking your code automatically. Please use this program before submitting your code so that you can catch obvious compilation errors, output errors, source code management errors, etc.

Tester Setup

You will need to setup your path to make it simpler for you to access the tester. To do so, you should check whether you use `csh` or `bash` as follows:

```
% echo $0
```

If the output shows `sh` or `bash`, then you are using `bash`. If the output shows `csh` or `tcsh`, then you are using `csh`.

1. For `csh`, add the following to the end of your `~/.cshrc`:

   ```
   set path=( /cad2/ece353s/tester/bin $path)
   ```

2. For `bash`, add the following to the end of your `~/.bashrc`:

   ```
   export PATH=/cad2/ece353s/tester/bin:$PATH
   ```

Then make sure to *log out and log back in*. Run `echo $PATH` and you should see the new path.

Testing your code

Please use the following commands after you have compiled and run your code. For example, for Lab 1, you should write, compile and run the various simple programs described in the lab handout, and then run the tester.

```
% cd ~/ece353/warmup
% ece353-tester 1
```

You will need to be in the directory associated with the lab (e.g., `warmup` for Lab 1, `threads` for Lab 2 and 3), and specify the lab number for which the tests should be run. For example, for Lab 1, the `lab_nr` is 1. For each test, you will see one or more PASS or FAIL results.

You can see the output that is *expected* by the tester by specifying the `-v` (verbose output) option to the tester. Note that the tester looks for regular expressions, not exact matches. Please use `ece353-tester -h` for more details. Please see the **tester FAQ (https://q.utoronto.ca/courses/419441/pages/faq-tester)** also.

The output of the tester will be stored in two files in the directory in which the tester is run:

`tester.log`: Shows the output of your program when running the ece353 tester.

`tester.out`: Summarizes the marks for the various tests.

Simulating Marking

The tester helps you test your code. However, you need to simulate the marking functionality of the tester to find out whether you have submitted your code correctly.

Before you do this step, make sure to **submit your code (https://q.utoronto.ca/courses/419441/pages/submitting-lab-code)** , or else the marking functionality will not work correctly.

Then, you can simulate the marking functionality by adding the `-m` option to the `ece353-tester` command:

```
% cd ~/ece353
# make a brand new directory
% mkdir marker
% cd marker
% ece353-tester -m assignment_nr
```

The `-m` option will simulate the marking functionality. It will checkout your code from your remote repository (which you have created as part of submitting your code) using the assignment's end tag (e.g., `Lab1-end`), build the code, and then run the testing functionality on this code. Note `assignment_nr` is the lab number (1, 2, etc.).

The output will be stored in three files, shown below, in the `~/marker` directory where you ran the tester. The XXX shown below is your user number. The last two files are copies of the original outputs of the tester.

`marker-XXX.log`:	Shows the output of retrieving files from your remote repository, building your code, and running the marker functionality.
`tester-XXX.log`:	Shows the output of your program when running the ece353 tester.
`tester-XXX.out`:	Summarizes the marks for the various tests.

We will run the same marker program to mark your lab. You will receive these files by email when we finish marking your lab. Let us know if you don't receive these files within one week after the lab submission deadline.

We do not use any "hidden" test cases, so there should be no surprises. Make sure to run the marker before submitting your code - it will tell you the marks you will get for your submission! **There are two exceptions to this rule:**

1. If your code has bugs and provides correct answers in some runs, and wrong answers in other runs, then it is possible that when we mark your program, it doesn't provide the same answers as when you ran the marker. This is especially possible with threaded programs that may occasionally run incorrectly due to concurrency bugs. Thus, for some tests, our marker runs your code three times and your mark is based on the minimum obtained in these three runs.
2. If your code is performance sensitive, e.g., it occasionally runs slowly, then it is possible that when we mark your program, then it will not meet our performance expectations. Normally, this should not be a problem since we will run your code on an idle machine and so your code will not be affected by other programs.