Objectives - Thu 3/31/2022 [1]

- Unit testing
- Test code coverage
- How to think of test cases

--

[1] The date of this lecture might be incorrect.

3 A's

- Arrange
- Act
- Assert

Also known as the "AAA (Arrange-Act-Assert)" pattern

Unit testing

Order

- Tests should be able to run in any order
  - Ex: test_read(…) should not depend on test_write(…)
  - It shouldn't matter if you run…

    ```
    mu_run( test_write );
    mu_run( test_read );
    ```

    … or …

    ```
    mu_run( test_read );
    mu_run( test_write );
    ```
- You should be able to comment out some tests without affecting others
  - Normally, you should be running all tests together
  - Need enough support code so each test is indepdendent.
- Every test should start with a clean slate

No manual inspection required

- The tests should be able to run on their own
  - Running all tests should require no human effort.
- This is the foundation of regression testing
  - Regression testing means running all tests whenever something changes and/or periodically (e.g., nightly).

Bugs vs. run-time error handling

- "Bugs" are flaws in <u>your code</u>.
  - Ex: You forgot to check for something.

- "Run-time error handling" means ensuring that the program behaves in a way that is helpful to the user, even when it receives unexpected or malformed inputs
  - Ex: malformed BMP header

Types of test code coverage

- "Line coverage" means every line of the code being tested was executed at least once.
- "Branch coverage" means for every conditional jump (If/While/For/Switch), we took the jump (condition true) and did not take the jump (condition false) at least once.
- "Path coverage" means we tested every possible path through the code (unique combination of branches). *This can be hard.*

line coverage $\subseteq$ branch coverage $\subseteq$ path coverage

```
////// IMPLEMENTATION CODE //////
void report_weather(bool is_sunny, bool is_raining) {
    if(is_sunny) {
        printf("The sun is shining.\n");
    }
    else {
        printf("The sun is not shining.\n");
    }

    if(is_raining) {
        printf("It is raining.\n");
    }
}

/////////// TEST CODE //////////
void test_report_weather_1() { // LINE coverage
    report_weather(true,  true);  // The sun is shining.  It is raining.
    report_weather(false, true);  // The sun is not shining.  It is raining.
}

void test_report_weather_2() { // BRANCH coverage
    report_weather(true,  true);  // The sun is shining.  It is raining.
    report_weather(false, false); // The sun is not shining.
}

void test_report_weather_3() { // PATH coverage
    report_weather(true,  true);  // The sun is shining.  It is raining.
    report_weather(true,  false); // The sun is shining.  It is raining.
    report_weather(false, true);  // The sun is not shining.  It is raining.
    report_weather(false, false); // The sun is not shining.
}
```

“Support functions” vs. “Helper functions”

For purposes of HW12 in ECE 264 (Spring 2019):

- “Support function” is used much like a helper function, but may be tested by external code (i.e., for the homework)
  - set_pixel(…) and create_bmp(…)
  - Note: “Support function” is not standard terminology.
- “Helper function”
  - _[redacted](…)
  - Not expected to be accessed by any external code.
  - This is standard terminology.

Thinking of test cases

- Easy cases
  - Answer is obvious (to you). If the test fails, you should have no doubt in your mind about whether the test itself is correct or not.
  - Ex: print_integer(5, 10)
- "Edge cases" (boundaries)
  - Extreme values for inputs (e.g., parameters, input files, etc.).
  - Ex: print_integer(INT_MIN, 10)
- "Corner cases" (turning points)
  - Look for **`if(▒){…}`**, **`while(▒){…}`**, **`for(▒){…}`**, and **`▒?▒:▒`** in your code
  - Will be captured whenever you have 100% branch coverage (hard)
  - Ex: print_integer(0, 10); print_integer(10, 16); print_integer(9, 16);
- Special cases (look for "except" in spec)
  - Look for words like "… except when…" or "Note: If …" in the specification.
  - Ex: mintf("%")

Note: This is not standard terminology. These are the instructor's invented terms.