TEST DRIVEN DEVELOPMENT (TDD)

ACKNOWLEDGEMENT: THE SLIDES ARE PREPARED FROM SLIDES PROVIDED BY NANCY M. AMATO AND JORY DENNY
SOFTWARE DEVELOPMENT

1. Solve
2. Implement
   1. Write test
   2. Write code
   3. Repeat
3. Integrate
4. Release
TEST DRIVEN DEVELOPMENT (TDD)

(Re)Write a test

Check if the test fails

Test(s) fail

Write production code

Run all tests

All tests succeed

Clean up code

Test succeeds

Repeat

1. Red
   - Refactor the code
   - Coherence is ensured by the unit tests
   - Write a unit test
   - Validate that the test fails

2. Green
   - Write just the minimum code necessary
   - Validate that the test now passes successfully

3. Refactor

ALL CODE IS GUILTY UNTIL PROVEN INNOCENT
In a shop, the shopkeeper wants an additional feature by which he can view the average sales for each month. The application he is using stores the total sales amount per day for each month in an array. How will you provide the additional feature?
How can we solve this?

Compute the average of the array
TEST DRIVEN DEVELOPMENT

- Write the test first
- Code needed to test your problem

```cpp
template<typename T>
T avg(T* arr, size_t sz) {
    return inf; //note this clearly does not work and is thus failing
}

void test_avg() {
    double arr = {4, 2, 3, 2, 3, 6, 8};
    double tolerance = .0001; //Can't compare doubles with strict equality
    double error = fabs( avg(arr, 7) - 4 );
    if(error >= tolerance)
        cout << "Test failed!" << endl; //This should execute for the first time
}
Before we continue, let's review

- Positives
  - Scaffolding, function interface, and test all implemented
  - We know it is good design
  - Tests to tell if the code is correct, before we struggle with debugging many lines of code

- Negatives
  - Code isn't written until later.....but is that really that bad? NO

In fact, with TDD you code FASTER and more EFFECTIVELY than without it
TEST DRIVEN DEVELOPMENT

- Now write the code and test!

```cpp
template<typename T>
T avg(T* arr, size_t sz) {
    T a = 0;
    for(size_t i = 0; i < sz; i++)
        a += arr[i];
    a /= sz;
    return a;
}
```
THINGS TO REMEMBER

- Always have code that compiles
- Test writing is an art that takes practice (and more learning!)
- Compile and test often!

Regression:
"when you fix one bug, you introduce several newer bugs."
Many frameworks exist, such as CppUnit, JUnit, etc.

We will follow SeTT:
- Setup
- Execute
- Test
- Tear Down