

**Spring 2020, CSE 4283 / 6283 – Software Testing and Quality Assurance**  
**Assignment-2, 100 Points**  
**Due: 11:59pm, Thursday, Feb 06, 2020 – on Canvas**

**Objective**

Apply test-driven development (TDD) to implement a set of software requirements. Write unit tests to provide adequate coverage of a code-base using your chosen unit testing framework and test runner.

**Scenario**

You have the responsibility to build an application interface to support various types of decision-making tools for the app end-users. The users accessing the app come from a wide range of disciplines and have different questions that the app will respond to. The application will provide users with a list of functions that they can execute. You are to create an application that allows the user to select and run each function and receive a correct response from the program. The new development manager requires that you code the application using a test-driven development approach. We define a unit as a function or method in a class with a single responsibility. The project should be developed and maintained using a GitHub repository. You will develop a single application with a central interface allowing access to the functionality listed below.

**Requirements**

Command Line Interface - Develop a command line app that prompts the user to select a function to execute and allows the user to gracefully exit the app when desired. The menu should be displayed after each function (although a GUI is not required, you are permitted to create one) unless the user exits. For now, the app must have the following functionalities.

1. **Body Mass Index** - Input height in feet and inches. Input weight in pounds. Return BMI value and category: Underweight = <18.5; Normal weight = 18.5–24.9; Overweight = 25–29.9; Obese = BMI of 30 or greater (see formula linked in the Notes & Resources section).
2. **Retirement** - Input user's current age, annual salary, percentage saved (employer matches 35% of savings). Input desired retirement savings goal. Output what age savings goal will be met. You can assume death at 100 years (therefore, indicate if the savings goal is not met).

## **Project Report**

Write a report summarizing your efforts. Your report should consist of the following content:

- **Report should be Professionally Organized / Presented** (content should not solely be a list of bullets that delimit the required content below - write in prose / paragraph form). NO ORDER SPECIFIED - Include Names, NetID, and Github username on the first page of the report.
- **Discuss each function and associated tests:** Describe function and constraints or assumptions made (used to justify chosen boundaries etc). How do you know functions tests are complete / thorough enough? What testing strategies did you use? How did you apply them (explain)? Discuss logic / motivation for each test (1-3 sentences).
- **Detailed setup and execution instructions:** Include links to download (languages, environments), instructions to setup and execute the application. I should be able to execute your instructions on a Windows 7 or a Windows 10 machine. Be explicit. State which versions of tools used (e.g., python 2.7.x) and what environment/OS to run on.
- **Screenshot or report output showing all tests suites /& test passing:** Include additional documentation so that I understand the meaning/content of each screenshot.

## **Notes & Resources**

- BMI Formula - <http://extoxnet.orst.edu/faqs/dietcancer/web2/twohowto.html>
- **Remember**, unit tests serve as documentation - your tests should indicate the features of the program
- Will use GitHub flow or similar to manage the project (main pristine branch with feature branches for each function). Can use fork and pull request flow.

**Turn-in the report on Canvas.**

## **Grading**

TDD process conformance - Quality of unit tests (comprehensive, well-documented) – Professional presentation of report, grammar/style, assignment instructions followed - Required functionality implemented - Thorough test strategy for each function - Clear instructions to execute program - Consistency and organization of test fixtures.