**Software Testing Q&A: Assignment 2**

Name: Tanner Hess | NetID: tmh648 | GitHub: TanHess  
Bagley College of Engineering, Mississippi State University   
Professor Tanmay Bhowmik   
March 10, 2022

**Overview:**

This report is a review of the testing of the BMI app built for assignment 2 of Software Testing and Q/A. In this app there is 1 function to be tested: the BMI function. The exact details of the testing will be described throughout this report along with the results of the testing procedures.

**Functions**

**BMI(feet, inches, pounds):**

**Description:**

This function takes 3 inputs as shown: feet, inches, pounds. The function first converts the input pounds into kilograms, then combines the feet and inches into just inches and converts this number into meters. After this, the function uses the BMI equation (bmi=kg/meters2) to compute the BMI of the given parameters rounding to the nearest 0.1. The function then uses the calculated BMI to calculate weight class based on these parameters: BMI = underweight < 18.5 >= normal <= 24.9 > overweight <= 29.9 > obese. After weight class is determined, the function returns a tuple containing the calculated BMI value and weight class.

**Tests:**

This function is tested for each of the 4 weight categories it can return: underweight, normal, overweight, or obese. For testing, the Pytest framework is used (external Python testing library). Each of the weight categories has it’s own test function parametrized with the various test cases used. Test cases were chosen according to the Nx1 boundary testing technique (as that is what I used to test the function). The reason I chose the Nx1 (1x1 for these tests) testing technique is so that my test cases could catch a boundary shift issue if it arose. Figure 1 shows the test cases chosen for each category using the Nx1 method. Also note that epsilon is .1 for this function and these test cases so the boundary points are selected accordingly.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test Cases | 1 | 2 | 3 | 4 | 5 |
| Underweight | 0 | 0.1 | 12 | 18.4 | 18.5 |
| Normal | 18.4 | 18.5 | 21 | 24.9 | 25 |
| Overweight | 24.9 | 25 | 28 | 29.9 | 30 |
| Obese | 29.9 | 30 | 50 | N/A | N/A |

**Figure 1**

**Execution:**

**Example of a full execution**

**Text

Description automatically generated**

This section contains primarily screenshots demonstrating 2 things: the execution of the test file, as well as the tests running manually through the app interface along with the expected results. For the manual execution of the test cases, I will omit the redundant tests (where the test cases overlap: i.e. underweight: 18.4 and normal 18.4).

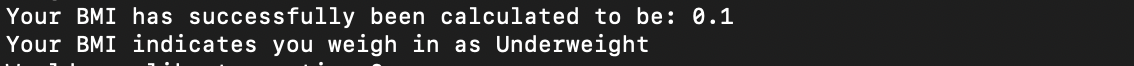
**Test file execution**

A screenshot of a computer

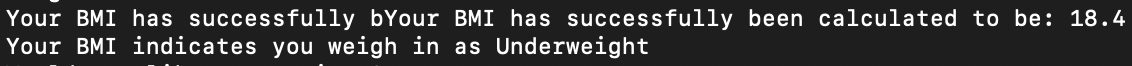
Description automatically generated

**Manual execution**

****

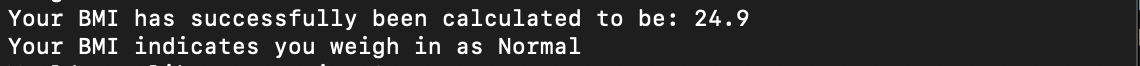
****

****

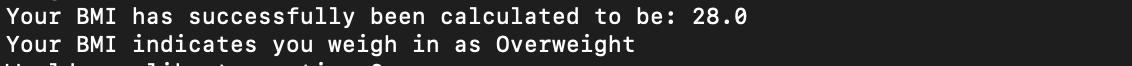
****

****

****

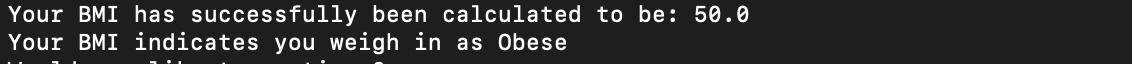
****

****

****

****

****

****

**Boundary Shift:**

For one execution of test cases, I introduced a boundary shift in my code. I changed the boundary at the lower end of normal and the higher end of underweight. The boundary shift changes the boundary bordering Underweight and normal from 29.9 to 29.8. The change in code can be seen in the next two screenshots.

Text

Description automatically generated

**Original Code**

**Text

Description automatically generated**

**Code Introducing Boundary Shift**

Below is the execution of the test cases after the boundary shift was introduced. As can be seen, the boundary shift is detected by the code. This is due to the implementation of Nx1 technique. This technique can determine a boundary shift problem where EPC would not have. The reason Nx1 was able to detect a boundary shift problem was because we used 1 point “on” the boundary, and one point “off” the boundary. The “off” point guarantees that we will be able to catch a boundary shift problem as regardless of the way the boundary shifts, either the “on” OR the “off” point will catch the mistake in code.

**Text

Description automatically generated**

**Instructions for Setup**