Coding Activity 7.1

Unit Testing Activity

#### **Learning Outcomes Addressed**

 1. Construct tests that validate the functionality of your code

 5. Write unit tests

Testing your application is a crucially important task. It ensures that the code you are writing is secure and performs as expected. There are multiple levels to testing an application, with some real-world examples:

* **Unit Testing: happens at the application code level; or, said another way, in the code used to write a standalone program or application. If you were developing a pen in the factory, this would be like testing that the cap, ballpoint, plastic case, and inkwell all work on their own.**
* **Integration Testing**: conducted to evaluate the compliance or interactions of a system or component with specified functional requirements. To use the pen example, this would be testing that the cap fits on the plastic container, the ballpoint integrates with the inkwell, and so on.
* **System Testing**: validates the complete and fully integrated software product, checking to see if the software works the say it is supposed to. With the pen, we’d test that the pen meets the functional requirements of being able to be held for long periods of time without discomfort, consistently has a good inkflow, and is able to write on several different types of paper.
* **Acceptance Testing**: evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery. For a pen, this would be pass/fail on if the pen’s functionality meets the specific needs it was created to fill. So, does it write, does it look a certain way, does it have a certain ink color, is it a certain weight, does it have the logo placed correctly, and so forth.

***Unit testing*** is the first level of application testing. As a developer, your focus should be on providing good unit testing to prevent any issues down the road. The more you progress in testing levels, the more expensive it becomes to solve application issues. Unit testing is the first line of defense to ensure an application is doing what it was designed to do.

In this activity, we will walk you through a list of requirements for a function. **Your task is to provide a list of possible Unit Tests that will cover these requirements.**For each step in the listed requirements, create the unit tests that you believe will capture the specific requirement.

You need to write the unit tests, and implement the function *Greet()*described below, then upload a screenshot of your work.

**Here are the requirements:**

* First, Write a function called *Greet(name)* that takes an argument called *name,*and returns a simple greeting message. For example, if the *name*is “Elizabeth” the function should return “*Hello, Elizabeth”*.
* Next, handle null values by introducing a default. For example, when the *name parameter* is *null*, then the method should return the string*“Hello there!”.*
* Next, add shouting. When “name” is all uppercase, then the method should shout back to the user. For example, when the name is "JOSE" then the method should return the string "HELLO JOSE!".
* Next, Handle two names as input. If the *name parameter*is an array containing two names, then both names should be greeted. For example, if the input parameter is [‘Jose’,’Pep’], the method should return the string: *“Hello, Jose, Pep”.*
* Finally, Handle an arbitrary number of names as input. If the *name parameter*is an array of multiple names, all names must be greeted. For example, if the input parameter is [‘Alex’,’Arsene’,’Jose’,’Zidane’], the method should return the string: *“Hello, Alex, Arsene, Jose, Zidane”*

*Here’s an example of a unit test that checks if a function exists:*

