**Module 4 Milestone**

Zane Milo Deso

Southern New Hampshire University

CS-405: Secure Coding

**Summary**

**Introduction**

As a Computer Science student enrolled at Southern New Hampshire University, I have been assigned a milestone assignment, which requires the implementation of 13-unit tests. The unit tests were to be completed using the Google Testing framework. This framework allows testers access to ASSERT\_ and EXCEPT\_ macros—making the process of running each test case straight forward. In this paper, I will outline my process for completing the tests, note any problems encountered, and explain the way my solution pathway for the problems.

**Unit Test Writing Process**

To complete each of the unit tests, I first read each TODO comment, noting to myself how I might approach each briefly. Now familiarized with the assignments overarching expectations, I began by reviewing the example test cases at the top of the provided code.

Following the review, I started coding each unit test in accordance with the corresponding TODO comments requirements. For example, one of the unit tests TODO comment asked me to create a unit test to verify the size of the collection vector after calling the addentries method. I was able to fulfill the requirement by utilizing the ASSERT\_TRUE macro to verify the collection size was equal to the size that would correspond with adding the number of entries I selected. Furthermore, I made sure to run all tests after each change to each subsequent test case. This approach proved invaluable as I would have spent more time debugging than working on the unit tests.  
**Problem Encountered: Greater Than Or Equal To**

While building most of the test cases went smoothly because each TODO comment seemed to provide simple test cases. I eventually ran into a small problem when I was attempting to verify collection capacity greater than or equal to some given value. The problem necessitated browsing through the Google Test macros, albeit I opted to use the implicit inclusive or (||) operator. As soon as I finished typing out the new expression, the linter cleared the flags. I ran the test with the unit tests behaving as expected. Although, I noticed that I could have used the ASSERT\_GT macro, I felt the expression was easier to read.

**Problem Encountered: ASSERT\_THROW**

For the final test case, I decided to attempt to use the ASSERT\_THROW macro, albeit I ran into a problem quickly. I noticed that no matter how I wrote the macro expression, the linter was constantly flagging the code. I opted to refer to an earlier test case using the same macro, which verified that I was accidentally using a method call that was not only improperly written, but also was the wrong type of expression to throw the exception I expected; a bad\_alloc exception.

To ameliorate the problem, I switched out the method call for the reserve method, which I ensured was the correct syntax, and used a very large integer to cause the bad\_alloc exception. I noticed that, no matter how large the number was either the exception was not being thrown or once the integer was too large for the int type, the linter flagged it. To get around this, I simply reduced the integer to a properly sized one and used multiplication operators with other integers to reach the bad\_alloc exception range.

## **Conclusion**

In conclusion, I have completed 13-unit tests utilizing the Google Testing framework by leveraging the various macros at my disposal, outlined briefly my unit test writing process, and explained what problems I encountered, ensuring to include the solution steps clearly. By closely following directions, reviewing example code, verifying test case to TODO comment requirements, I was able to complete this assignment to the best of my ability.

**Final Output Screenshot**

