**Module One Process Summary**

Joseph Veneski

Department of Computer Science, Southern New Hampshire University

CS-405 Secure Coding

Professor Kaan Esendemir

March 10, 2024

**Module One Process Summary**

**Overflow:** Overflow is the easier of the two cases to manage so that was the first goal. Using the template provided, all that needed to be done was to verify whether the result would result in an overflow and inform the user. A change was implemented to the add\_numbers template to return a tuple containing both the result, and a Boolean flag to inform of possible overflow. A simple check to see if adding the adding the increment to result would cause an overflow was added in an if-statement which flips the flag and breaks. A computer screen with text on it

Description automatically generated

This necessitated changes to the test\_overflow function to handle the return of a tuple instead of just the result. If-statements are again used to check whether the flag indicates an overflow, and prints output to user. A computer screen shot of a program

Description automatically generated

**Underflow:** Handling underflow was slightly more complicated than overflow. Again, the template was modified to return a tuple of both the result, and a Boolean indicator similar to the overflow changes. An if-statement is used to determine whether the decrement is greater than the result, OR the result minus the decrement would be less than the limit. The first part handles unsigned types as a greater decrement than result causes an underflow since it cannot go below zero. The second part is similar to the overflow logic, if the arithmetic occurs and it results in a value lower than the limit there is again an issue. In either of these cases the Boolean is flipped and the function returns. A computer screen shot of a program code

Description automatically generated

Like the overflow process, test\_underflow required updating to handle the tuple return. If-else statements are used to check the flag, and display output. If no underflow occurs the result is displayed and if underflow would occur the user is informed. Although there should be no overflow or underflow on the tests stating without, handling is implemented as good practice in case something unexpected occurs. A computer screen shot of a program

Description automatically generated

There were no issues identified during the static code analysis as shown using cppcheck: A screenshot of a computer

Description automatically generated

Console output of running the modified code: A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated