# **CS 405 1-3 Activity: Numeric Overflow Coding**

# **Austin Donaubauer**

# **Southern New Hampshire University**

# **CS-405 Secure Coding**

# **Wasim Alim**

# **July 1st, 2025**

# **CS 405 1-3 Activity: Numeric Overflow Coding**

The 1-3 Activity Numeric Overflow Coding assignment was a coding practice in secure coding for overflow and underflow of datatypes. We were given the code template to work from and had to add logic to output the results when a data type overflow or underflow occurs. I edited the add\_numbers function to return a Boolean value instead of the result. Returning a Boolean helps so that I could return true or false for different conditions. Those conditions included checking for the datatype as an integral using a C++ standard library function called is\_integral from the type\_traits library. This checks for integer datatypes like char, int, short, and long. After checking the type of datatype, I checked for the overflow or underflow using a check to see if adding to the result would go over the maximum or minimum values. To obtain the maximum and minimum values, I used the numeric\_limits library which has functions for max and min. If the result before adding the increment will exceed the max or min, I return a Boolean false to indicate the overflow or underflow. After checking the integral, I allowed the value to be incremented. Then checked for the floating\_point values, and then checked if the result was infinity using the isinf method from the cmath library. If it is infinity, I returned a Boolean false value to indicate that an overflow occurred. If the result passes the checks, then I assign the local result to the reference result variable, and then return a Boolean true to indicate no overflow or underflow detected. I used similar methods for the subtract\_numbers method by checking the datatype using the is\_integral and is\_floating\_point. Then checked if the result would go beyond the minimum or maximum to return a Boolean of false to indicate the underflow and overflow would occur. For this assignment, I found a challenge because for the underflow, a datatype like char would start at 127, and subtracting by the decrement resulted in -23 when subtracting 25 six times. This does not result in an underflow because the char type has a minimum of -127. To fix this and indicate a logical underflow for value below zero, I had to check for the signed datatypes which is the positive part of the datatype value and uses non-negative values. This check works so that if it find that the result value will be negative or less than 0, it will return a Boolean of false to indicate the underflow. After these checks, I allow the result to be decremented. Then checked for the floating point database and check for infinity and negative values to return a Boolean of false to indicate the underflow. Lastly, I assign the local result to the result reference, and then return true to indicate when there is no overflow or underflow

For editing the test\_overflow and test\_underflow methods, I added the result reference variable, an output statement indicating adding or subscription with or without overflow or underflow. Then check the result of the add\_numbers and subtract\_numbers and outputted the result if it returns true, or outputs an Overflow / Underflow detected statement to indicate to the user what occurs.

I created some debugging outputs when working with the code that I left in but commented out after thinking the output is correct for this assignment.

Overall, this assignment went well I think, and I look forward to learning more of these types of concepts and how to code for these types of issues. Below is the screenshots of the output of my program.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.