**1-3 Activity: Numeric Overflow Coding**

**Ryan DeBraal**

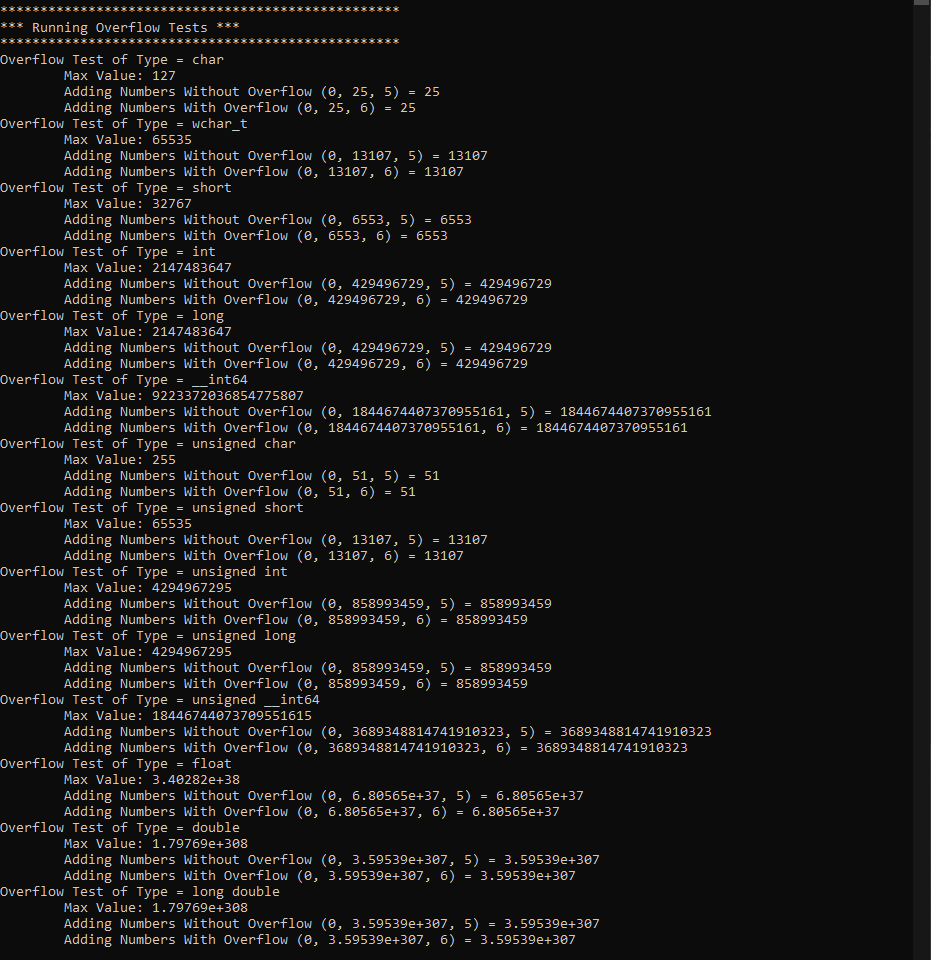
**CS-405-X6389 Secure Coding 21EW6**

**Summary**  
  
In order to complete this assignment correctly it is important that we do *not* use the product of (increment/decrement \* i) in an if statement because that calculation itself would result in an overflow/underflow.

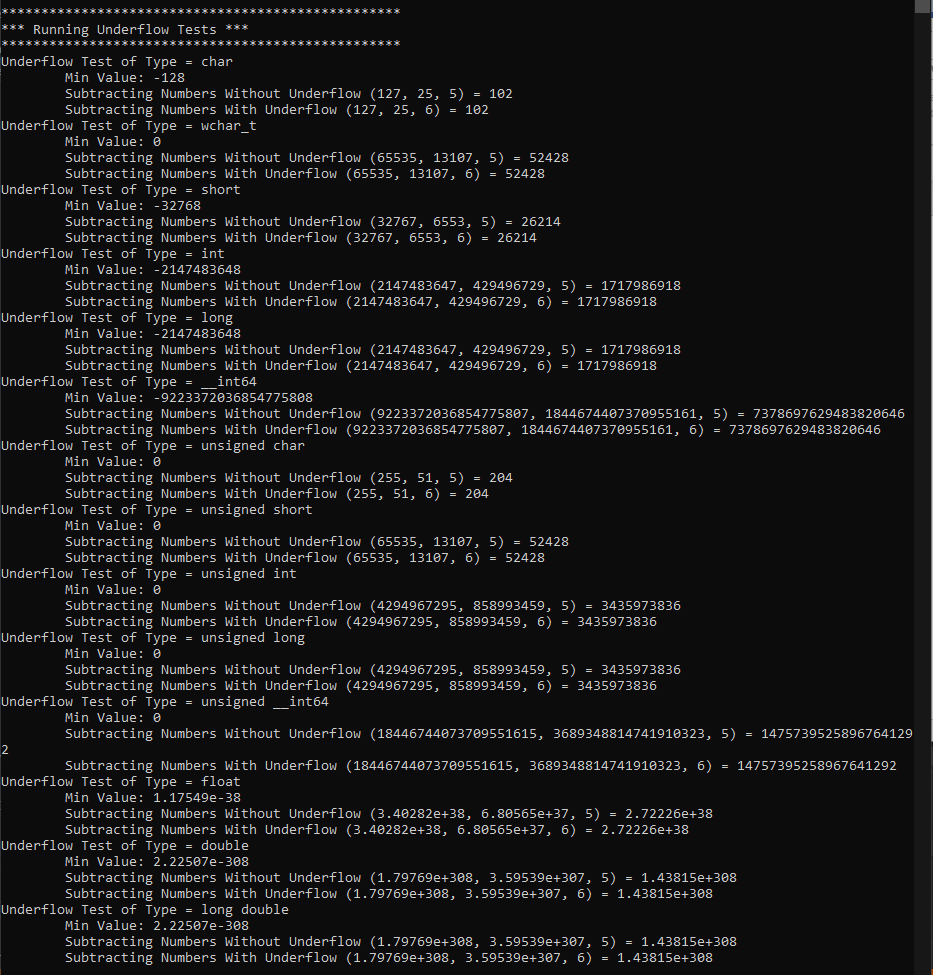
It only *appears* to work because the compiler automatically casts the result of (increment/decrement \* i) to a larger data type, but the flaw in this strategy is exposed when we try to test against an "unsigned long long" because it does not have a larger numeric data type available in which to upcast. When the value goes beyond the range of its data type it results in “wrap around” (sometimes called “modulo wrapping”, or more obscurely, “saturation”).

“An integer overflow or wraparound occurs when an integer value is incremented to a value that is too large to store in the associated representation. When this occurs, the value may wrap to become a very small or negative number. While this may be intended behavior in circumstances that rely on wrapping, it can have security consequences if the wrap is unexpected.” (Plover, 2021)

**Overflow tests**



**Underflow tests**



**Citations**

Plover, (2021, March 15). Common Weakness Enumeration. CWE. <https://cwe.mitre.org/data/definitions/190.html>.

**Resources**

<https://stackoverflow.com/questions/199333/how-do-i-detect-unsigned-integer-multiply-overflow/1514309#1514309>

<https://stackoverflow.com/questions/19842215/wrap-around-explanation-for-signed-and-unsigned-variables-in-c>

<https://www.learncpp.com/cpp-tutorial/unsigned-integers-and-why-to-avoid-them/>