Hanah Deering

1-3 Activity: Numeric Overflow Coding

CS 405 Secure Coding

Southern New Hampshire University

**Screenshots of the Application Console Output**

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generatedGraphical user interface, text, application, email

Description automatically generated

**Brief Overview**

To stop either the overflow or the underflow, I had to change a couple of things within the source code.

In the add\_numbers command, I added an *if-else* statement for when an overflow occurred. If both start and increment end up being more than or less than 0 and the result ended up being greater than zero or less than std::numeric\_limits<1>::max()), then an overflow would occur without the *if-else* statement. When the overflow occurred, it would give us a return value of -1, which is not ideal. With *else*, any other results aside from the two previously stated, would **not** result in an overflow and the result will be printed.

In the subtract\_numbers() command, I also added an *if-else* statement for any kind of underflow that may have occurred. If both start and decrement were greater than or less than zero and the result was greater than or less than std::numeric\_limits<1>::min()), an underflow will occur, in turn making the return value -1. Adding the else portion made any other results pass without an underflow.