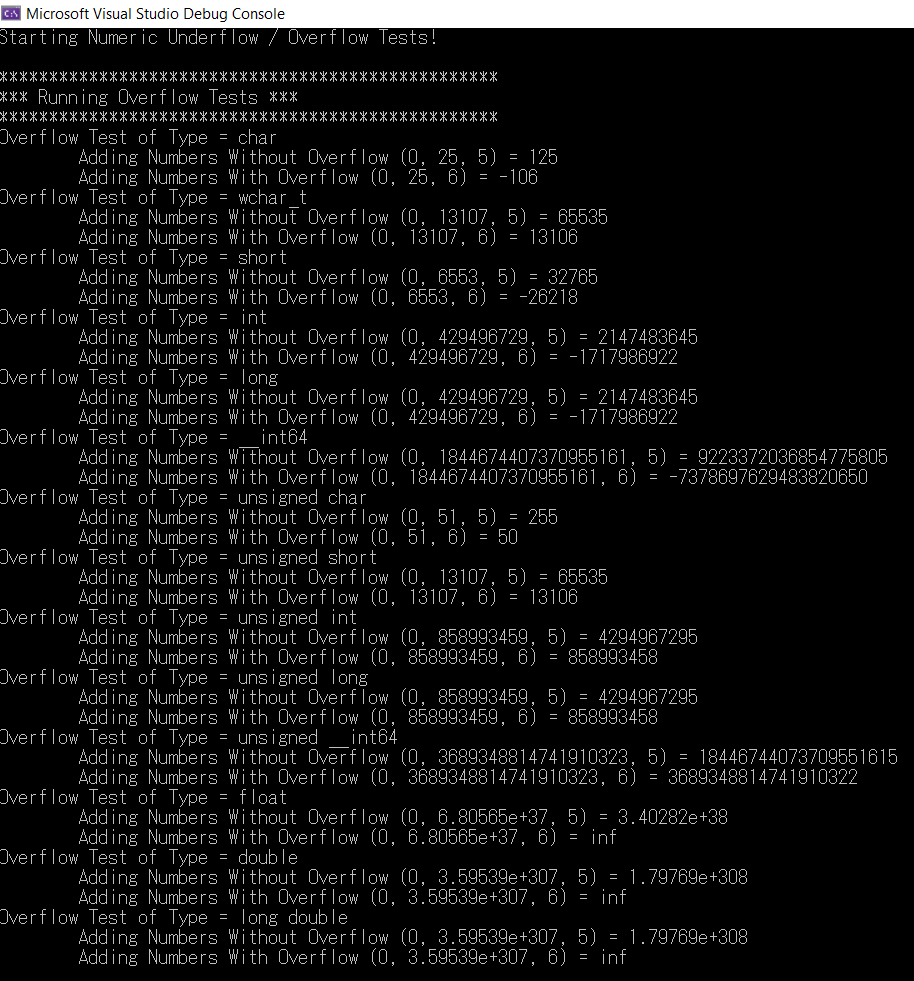
Joseph Silva Jr.

SNHU

CS 405: 1-3 Numeric Overflow

10/31/2021

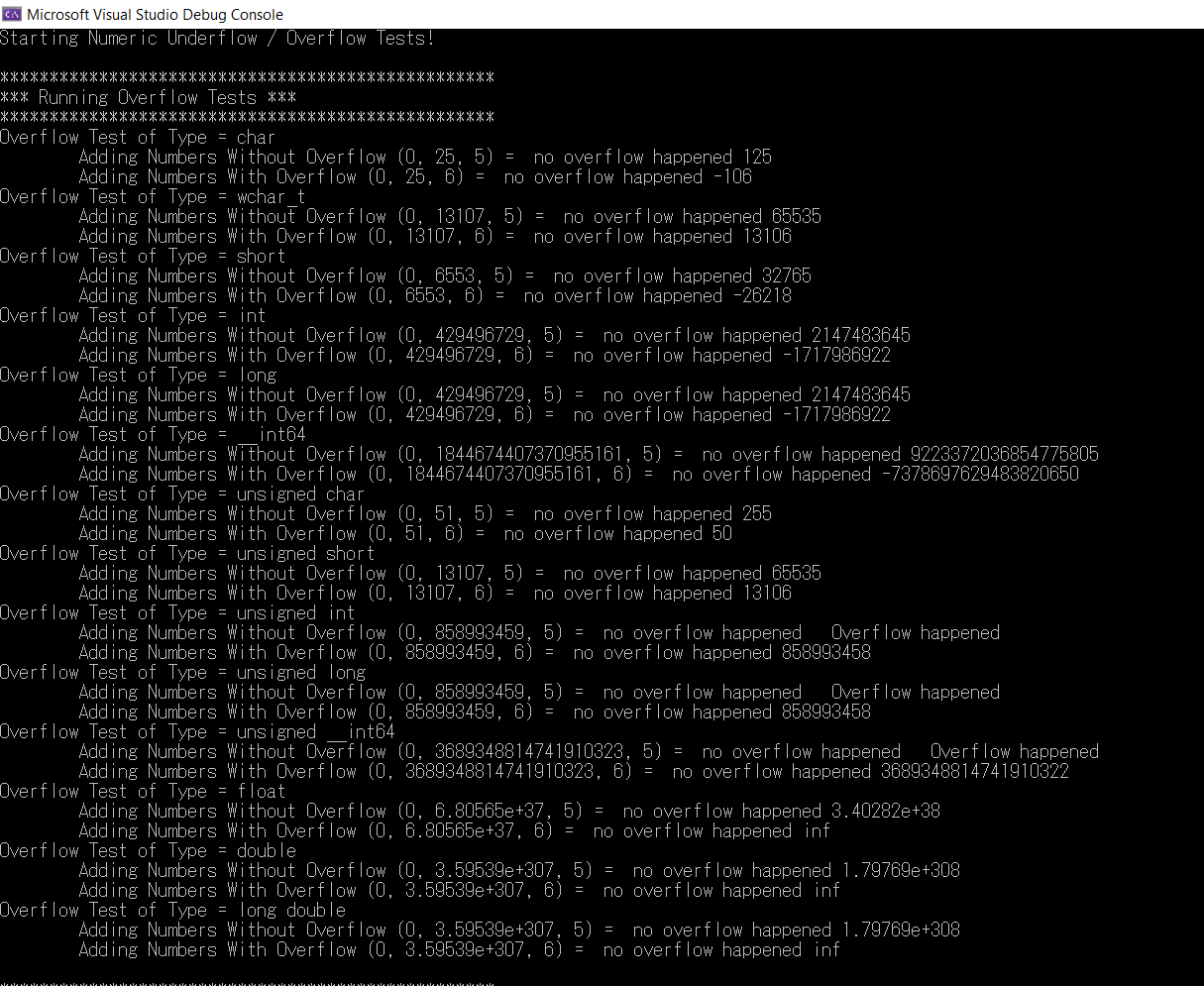
**Original Code Debug Results**



Text

Description automatically generated

**Updated Code Debug Results**



Text

Description automatically generated

**Summary:**

For the T add\_numbers() command, I added an if-else statement to show when an overflow occurred. First, if *start* and *increment* were greater than 0 and *results* is greater than std::numeric\_limits<T>::max()) then an overflow occurred. Next, if *start* and *increment* were less than 0 and *results* is less than std::numeric\_limits<T>::max()) then an overflow occurred. When an overflow occurred, the return value would be -1. Last, else any other results besides the two above, an overflow did not occur and return value was result. During the test\_overflow() if return value was -1, an overflow occurred, else the return value was result than an overflow did not occur and the result was printed.

For the T subtract\_numbers() command, I added an if-else statement to show when an underflow occurred. First, if *start* and *decrement* were greater than 0 and *results* is greater than std::numeric\_limits<T>::min()) then an underflow occurred. Next, if *start* and *decrement* were less than 0 and *results* is less than std::numeric\_limits<T>::min()) then an underflow occurred. When an underflow occurred, the return value would be -1.Last, else any other results besides the two above, an underflow did not occur and return value was result. During the test\_underflow() if return value was -1, an underflow occurred, else the return value was result than an underflow did not occur and the result was printed.