### Southern New Hampshire University

### CS 405 – Secure Coding

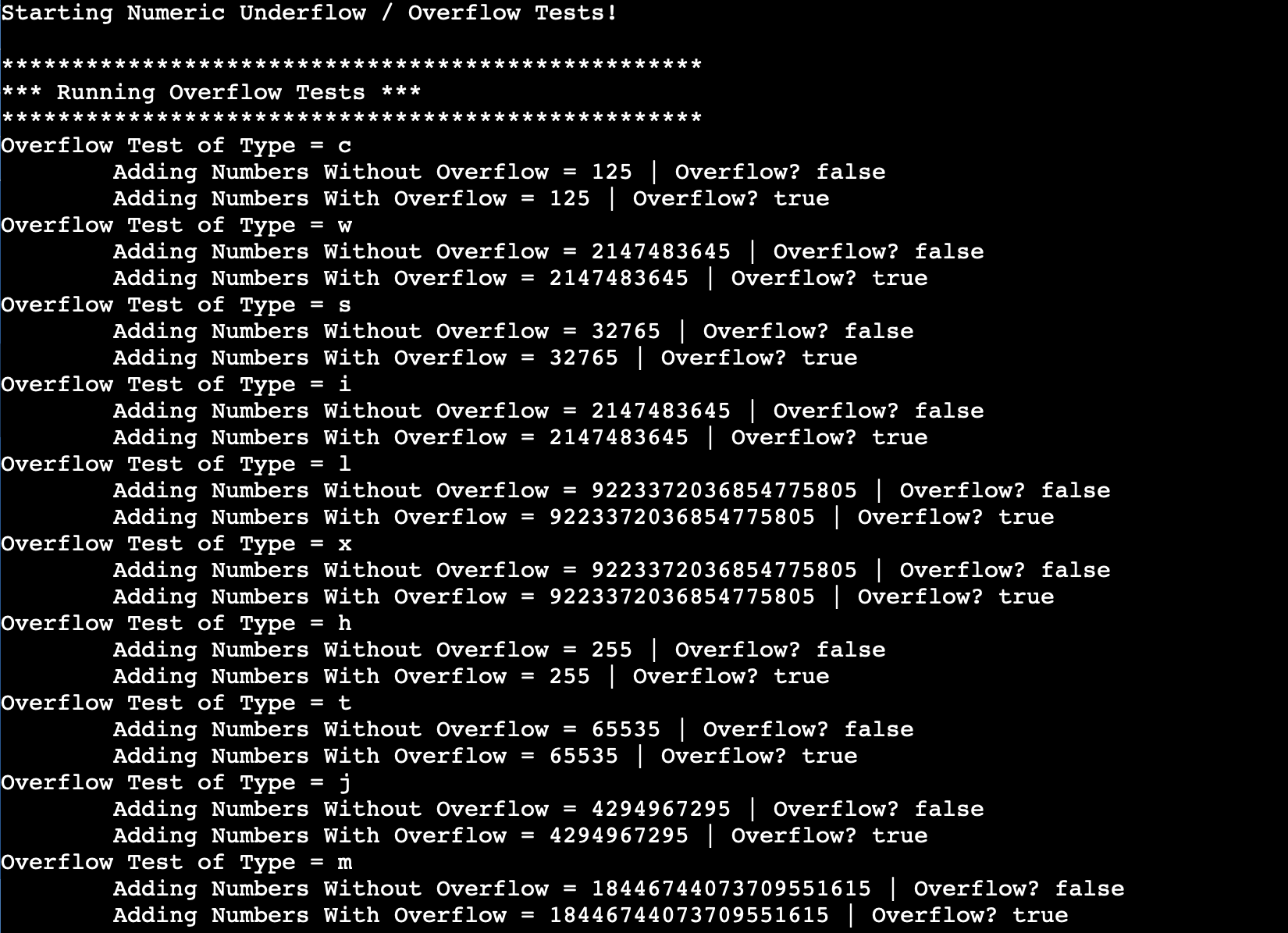
### 1-3 Activity: Numeric Overflow Coding

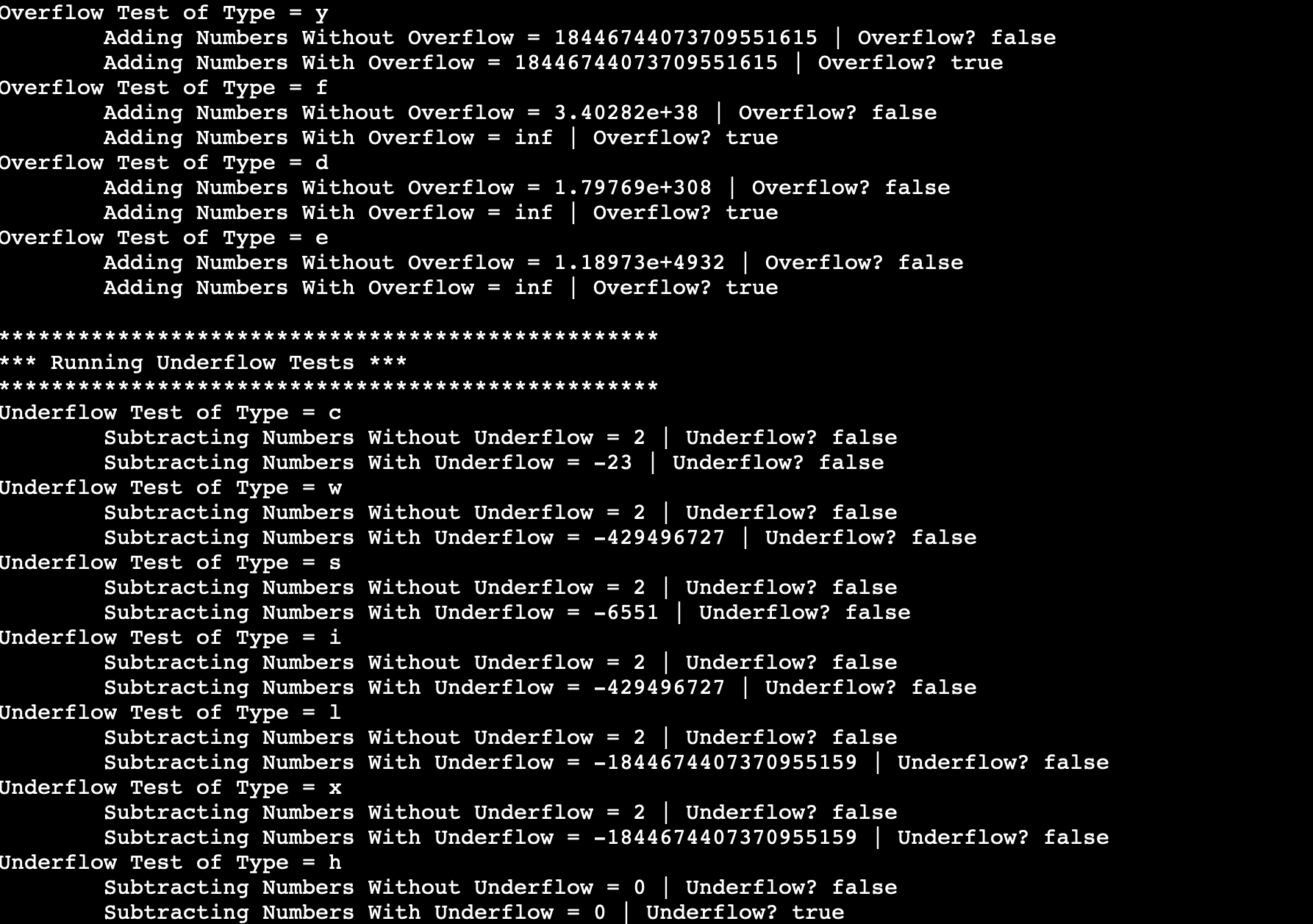
Kenneth Wilkerson

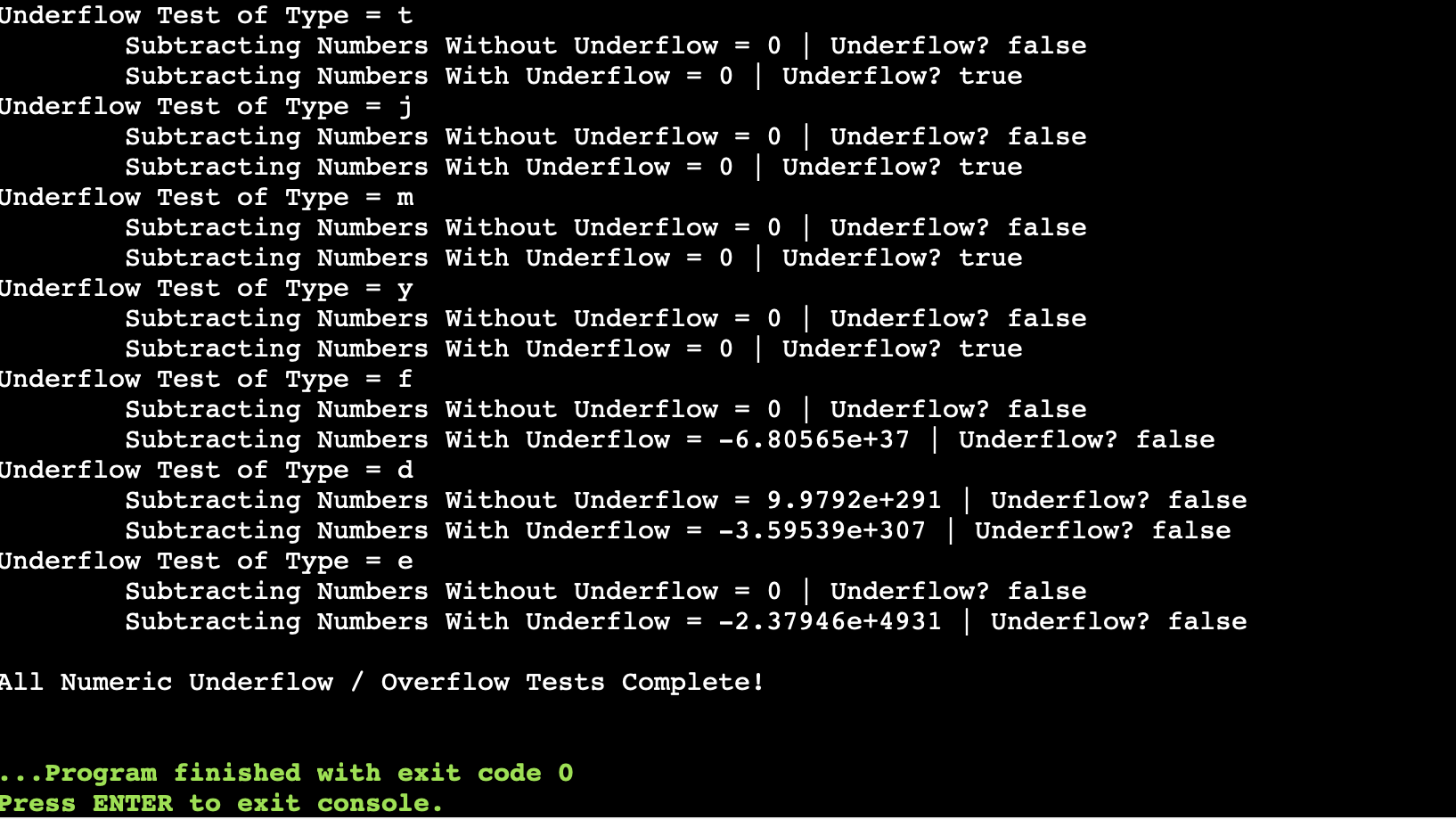
### Professor Andujar

September 7, 2025

**Screenshots**

****





**Process Summary**

In this assignment, I modified the functions to detect and prevent overflow and underflow for both integers and floating-point numbers. For addition, I added a pre-check for integral types to ensure that the next addition would not exceed std::numeric\_limits<T>::max(). For floating-point types, I used !std::isfinite() to detect overflow. I also added a boolean flag to report the overflow status. For subtraction, I implemented a pre-check to detect underflow in signed integers and to prevent wrap-around in unsigned integers. Similarly, !std::isfinite() was used for floating-point detection and a boolean flag was added to report the underflow status. The test functions were updated to display the numeric result and the overflow/underflow status. I also made sure that the templates consistently across signed, unsigned and floating-point types. While coding I encountered a few challenges. Initially, signed integer underflow was not being detected because using lowest () + decrement caused a wrap-around before the check. I resolved this by implementing a pre-subtraction comparison (result - decrement > result) that detects underflow. For unsigned integers, I also needed to add a check (result < decrement) to avoid wrap-around. After addressing these issues, the program now successfully detects, prevents and reports numeric overflows/underflows for all tested types.