This assignment is designed to add to a program's resilience by addressing the numeric overflow & underflow. I started by modifying the core functions of the code, using “add\_numbers()” & “subtract\_numbers” while proactively checking using “std::numeric\_limits.” These validations detect whether an operation would exceed the data type’s capacity before allowing it to execute, preventing any undefined behaviors or potential data corruption. If a risky operation is identified, the functions should now safely halt it’s execution & return the appropriate maximum or minimum values. By providing clear console messages for the user, we are able to remain informed as these changes occur.

TO ensure validation, I also updated the corresponding test functions using “test\_overflow()” & “test\_underflow()”. This aligns with the new logic, allowing my tests to recognize & report prevented operations as successful outcomes to improve it’s safeguards. These improvements should allow for more secure code, drawing on the use of the standard library.  
  
\*See console photos below\*  
  
