In this project, I took the existing C++ program and made some changes to handle numeric overflow and underflow during addition and subtraction. I added checks in the “add\_numbers” and “subtract\_numbers” functions to catch any overflows or underflows before they happen. Basically, I compared the results of the calculations with the limits of the data types to see if they were going to go out of bounds.

To prevent issues, I updated the functions to return two values: one tells whether there was an overflow or underflow, and the other gives the result. If something goes wrong, the program lets the user know, so they’re not left guessing what happened.

I also tweaked the testing functions to make sure they inform the user whenever an overflow or underflow occurs. This way, the console output clearly shows the status of each operation, which makes debugging a lot easier.

I tested the program with a bunch of different data types, including signed and unsigned integers, as well as floating-point numbers. The output is captured for submission, showing that the program now does a good job of identifying and handling numeric overflow and underflow situations.

A screenshot of a computer program

Description automatically generated