For the activity, I detected overflow and underflow by getting the minimum and maximum values of the provided datatype by using numeric\_limits<T>::max() and numeric\_limits<T>::min(). For overflow, within the for loop, if the current number **result** was greater than the data type’s maximum value – increment, then an overflow is about to occur. The logic behind this is a maximum number minus the increment gives you the maximum value that **result** can be without causing an overflow. For underflow, the same logic is used if the current value of **result** is less than minimum number + decrement, and underflow will occur. To tell the calling test\_overflow and test\_underflow functions about whether or not an underflow/overflow occurred, I passed the address of the status variable to the add & subtract\_numbers functions that are then written to with an integer representing OK, OVERFLOW, and UNDERFLOW (0, 1, and -1). After the operation has concluded, I then call the print\_status() function which will output whether or not the operation was successful, or if an overflow/underflow was detected.

In the output for underflow tests, I noticed for char, short, int, long, and \_\_int64, an underflow was not detected. To the best of my knowledge, this is an accurate representation of the ongoing operations since it seems the resulting number accurately represents a successful subtraction.

Outputs:

Text

Description automatically generated

Text

Description automatically generated