CS 405 Module One Numeric Overflow Activity Guidelines and Rubric

Overview

You are a senior software developer on a team of software developers who are responsible for a large banking application. Your manager has learned that the code used for addition and subtraction over a series of steps will sometimes calculate incorrect results. The other developers have spent a lot of time tracing down the issue, and they narrowed it down to two functions: add_numbers() and subtract_numbers(). Following some guidance from an older testing guide, they isolated those functions in a standalone set of test cases. Doing this allows them to reliably show how to trigger an underflow or overflow. However, they don't know what to do to prevent it from occurring in the future. They came to you for help.

You will learn to do the following:

- Detect when a numeric overflow or underflow is about to happen
- Write code to prevent the numeric overflow or underflow from happening
- Protect your code from underflow and overflow in any of the standard C/C++ data types

Prompt

This assignment presents code that is designed to show the two functions operating with and without overflow or underflows. Using the existing source code, you will add logic to the add_numbers() and subtract_numbers() functions to detect, prevent, and notify the caller of a numeric overflow. You will also modify the calling test_overflow and test_underflow functions to be notified of success or failure of the add or subtract functions they call, and to modify them, the functions write to the console with overflow status (true or false) and the numeric result of the function.

The following are a few key notes:

- The source code has been commented with TODOs to explain the detailed rules you must follow.
- There are comments that mark code that must be changed.
- There may be more than one way to solve this problem, so be sure to demonstrate that you can detect an underflow or overflow, prevent it, and communicate it back to the calling function.
- Remember to leverage capabilities provided by the standard C++ library to help you achieve success.

Please comment on any changes you make in the code to explain the logic, formulas, or data types you are adding. You will also create a brief written summary of the approach taken. It should explain how this approach is designed to stop the overflow or underflow, any issues you encountered, and how you resolved those issues.

Specifically, you will prepare the following:

- Numeric overflow and underflow secure coding
- C/C++ program functionality and best practices
- A summary of your process in a Word document that contains a screenshot of the application console output

To complete this assignment, download the Numeric Overflow source code to use as guidance as you move through the activity. You will use your development environment to complete this activity.

What to Submit

Submit the modified source code as a ZIP file. Also include a Word document that contains a complete screenshot of the application console output and a brief summary of your process.

Module One Numeric Overflow Activity Rubric

Criteria	Proficient (100%)	Needs Improvement (75%)	Not Evident (0%)	Value
Numeric Overflow and Underflow Secure Coding	Completes the coding activity by successfully preventing numeric overflow and underflow for all data types; notifies the calling function; displays the overflow condition to the user	Completes most of the coding activity, but needs to address one or more of the following: does not prevent both overflow and underflow; does not work for all data types; prevents the underflow or overflow, but does not notify the calling function; does not display the result of the overflow to the console	Does not attempt criterion	70
C/C++ Program Functionality and Best Practices	Demonstrates industry standard best practices, including in-line comments and appropriate naming conventions to enhance readability of code; develops	Shows progress toward proficiency, but with errors or omissions; areas for improvement may include implementation of software design patterns in code	Does not attempt criterion	10

	functional C/C++ code that illustrates software design pattern approach			
Process Summary	Provides a summary of the debugging that is thorough and systematic, including specific types of bugs; accurately describes the corrections and provides screenshots of the application console output, from the compiler, and of the error list	Provides a summary of the debugging, but the summary lacks detail, the approach is unsystematic, specific types of bugs are not differentiated, the corrections are not fully described, or a screenshot is missing	Does not attempt criterion	20
			Total:	100%