The Calculator Problem

A classic example of a problem that uses Stacks is the calculator problem. In this problem, you will write the engine for a calculator that processes 'post-fix' commands. The traditional method of representing math statements is called 'in-fix' notation, such as:

2 + 2

However, a post-fix version would be:

2 2 +

In the post-fix scheme, the operation comes after the two numbers involved, not between them. The post-fix notation was common for a time in some calculators but is not common in computing as we learn in-fix notation, and that is what is familiar. The post-fix approach lends itself well to avoiding confusion on operator precedence and as an example of using stacks.

The following website offers a spectacular visualization for the process of using stacks to process post-fix commands.

https://www.free-online-calculator-use.com/postfix-evaluator.html#

Use this website and the test cases provided with this assignment to build your stack version of the post-fix calculator engine.

Your task is confined to a single file, Calculator.cpp. The existing code provides an input of a pre-processed vector of calculator commands. The vector contains a list of strings pre-filtered for any invalid inputs (e.g., non-numbers or operators). It ***does not*** validate if it is a working post-fix calculator command. For instance, an input vector could contain "2", "+", "2", but the vector is not valid as a post-fix command. If the command is invalid, your code will return a predefined error message.

The basic main method will run the test cases and report the results first. Once the test cases complete, the program calls the command-line Calculator for any additional testing.