Description:

You are to write a C++ program using stacks and linked lists. You may not use std::list, or any other C++ library list type structure to store your data. You must create and manage the nodes yourself. Do not create a separate list class.

Program Description:

First, you will write a program using the stack class that provides unlimited undo and redo capabilities for a simple text calculator. The calculator will be integer-only. The user will enter a command possibly followed by a positive integer. The calculator will perform the appropriate operation and report the current value of the calculation.

The possible commands will be +,-,*,/,%,C,U, R and Q. The arithmetic operations will be followed by an integer (with or without a space between command and number) and represent the appropriate integer calculation. C means clear. It will set the current value to 0. U is undo. R is redo. Q is quit. Use ">" to prompt for the next command. Redo will print the operation that is being redone as well as the result. If there's nothing to undo or redo, report that and then the unchanged current value.

At the beginning, provide a brief set of instructions to the user (be sure to match the sample precisely).

Note that you will need two different stacks to accomplish the task.

Sample partial program run (user input in bold):

$ ./a.out

Welcome to the Undo-Redo Calculator!

This calculator handles the following commands:

+, -, *, /, %, C, U, R, and Q

Arithmetic operators are followed by a single positive integer.

C will clear memory, U will undo, R will redo, and Q will quit.

0

>+300

300

>-500

-200

>/ 100

-2

>R

No operations to redo

-2

>U

-200

>U

300

>R

Redoing: -500

-200

>C

0

>U

-200

>R

Redoing: C

0

>U

-200

>+ 3

-197

>R

No operations to redo

-197

>Q

Goodbye

Make sure that your output matches the sample output exactly. Don't forget to use the diff tool in Linux (and possibly the VSCode file comparison).

To run the stack program with sample input and output (assuming you've copied the necessary files over and compiled) run the following commands:

./a.out < stackInputFile.txt > myStackOutputFile.txt

diff stackOutputFile.txt myStackOutputFile.txt

If your program works as expected, the diff should not output anything.

Linked List Stack Implementation:

Once your program is working, you will modify the stack class to use a linked list representation. You will convert it from using a static array (and being of fixed maximum size) to using a singly linked list and handling any amount of data, but you will not make changes to the public interface of the class. Make sure that you provide a correct destructor, copy constructor and assignment operator that use private helper methods to avoid code duplication. You will have to add those to the .h file, because the current class doesn't use dynamic memory and therefore relies on the default methods.

Use the stack tester program to test your stack class. Also use valgrind to help ensure that your stack class does not contain memory leaks.

Make sure that your program works correctly with either the provided Stack class or your linked list program, testing using diff and also making sure valgrind reports no errors.

Submission requirements:

The names of your stack program files must all begin with Stack. Name your files with clearly meaningful names (no StackMain.cpp, please). Zip your source code files into a single zip file (do not zip folders, just the files). Only include the files you modified or wrote (not OperationData.h).