

Lab 2: Vectors Part 2

CSCI 41

Objectives:

- Implement The Big 3 for the Vector ADT
- Use our custom Vector ADT to actually do something

Part 1: Get the starter code

First, `cd` into the folder where you want to store your class files (e.g., `~/labs`). Copy the starter code there with the following line:

```
gimme lab02@csci41
```

This will make a `lab02` folder in whatever directory you were in.

Part 2: Complete `vec.cpp`

Implement The Big 3® as well as copy over your previous code from last lab.

Part 3: Test your code

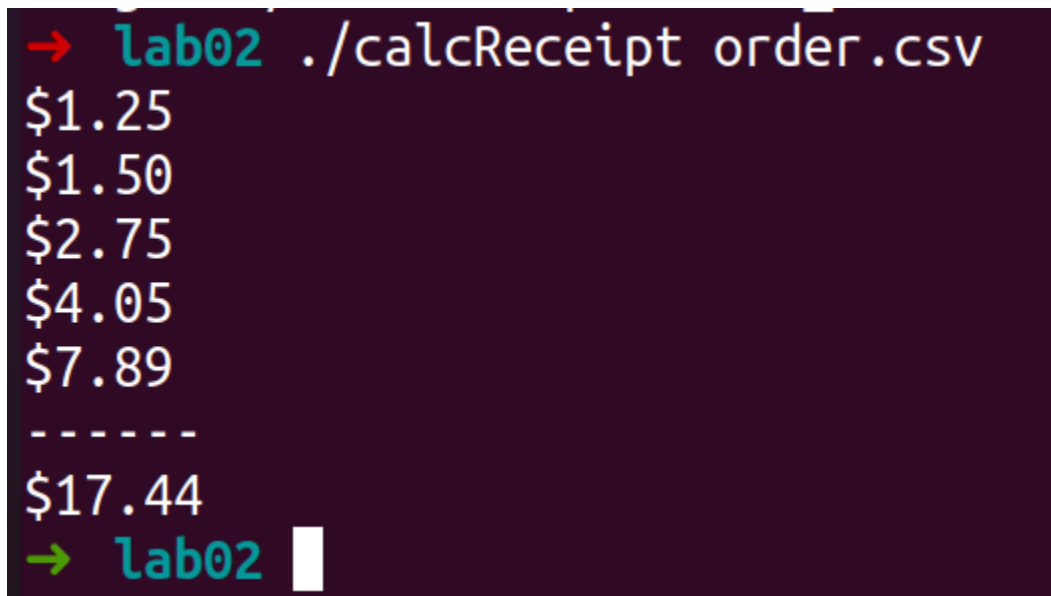
`testVec.cpp` holds some tests—work to get those passing. Then, you'll create 4 more tests that test some subset of The Big 3 in some way. Get those to pass as well.

To compile, use `make testVec`. See the Makefile for more details.

Part 4: Implement calcReceipt

calcReceipt.cpp contains a *use case* for our Vec class. In it, we read a file passed along in the command line arguments that represents a receipt—we'll extract the prices, sort them, and print them and their sum to the screen.

Take a look at the order.csv file. Each line contains a description, then a comma, then a price **in cents**. Next, look at the comments in calcReceipt.cpp. This is what the output of the program should look like when run as `./calcReceipt order.csv`:

A terminal window with a dark purple background. The prompt is a red arrow followed by 'lab02' in teal. The command './calcReceipt order.csv' is entered in a light blue font. The output shows five prices in white: '\$1.25', '\$1.50', '\$2.75', '\$4.05', and '\$7.89'. These are followed by a line of six dashes '-----' and then the total '\$17.44' in white. The prompt returns to 'lab02' followed by a white cursor block.

```
→ lab02 ./calcReceipt order.csv
$1.25
$1.50
$2.75
$4.05
$7.89
-----
$17.44
→ lab02 █
```

You will process each line to extract the price and add it to the vector. After you have all the prices, you'll sort the vector and calculate the sum. Finally, you'll print the information you've collected in a pretty way.

Hint: look into the `stoi` library function to convert a string to an int.

Compile using `make calcReceipt`.

Part 5: Submit your code

When you're satisfied with your solutions, you can submit them to the autograder for grading. **Do not submit and assume you got 100%—you may be unpleasantly surprised. Always check your grade.**

From your solution directory, submit your code to the autograder using the following command on the terminal:

```
turnin lab02@csci41 vec.cpp testVec.cpp calcReceipt.cpp
```

The autograder will grade your code within a minute or so. If it's not working, please yell at Lawton to fix it. Run `view-grades` to look at your grades on the terminal, or go to the class website to view them there. You may resubmit as much as you want before the due date—just follow this same process after you've updated your programs.

Rubric

Rubric Item	Points (34 pts total)
Your tests in testVec.cpp pass	8 pts (2 for each test)
My tests in testVec.cpp pass	10 pts (5 for each test)
There are no memory leaks when running my testVec.cpp tests (test yourself by running <code>valgrind --leak-check=full ./testVec</code>)	10 pts
<code>./calcReceipt order.csv</code> is correct	3 pts
<code>./calcReceipt order2.csv</code> is correct	3 pts