
Mathias Eriksen
mterikse@ucsc.edu
Lab 04
October 20 2022

In this lab, I did the work entirely on my own, and did not find it too difficult to implement the correct functionality given my previous coding experience. Within this lab, the most important concept was definitely the idea of a stack and how to use it. Knowing how to use the item indices, and how to access values within the stack were imperative in incorporating the stack functions. The lab asks us to implement an rpn calculator, which is a stack based calculator. In order to implement the correct behavior of an rpn calculator, you had to be familiar with string concepts, as well as think through the basic function of an rpn calculator. The final concept was that of error handling, and the most important thing here is to know what the error you are looking for is, and how to catch it correctly. This lab also required that one was quite familiar with loops and if statements in C, since they are implemented a lot.

My approach to this lab was to start by getting the stack functions working correctly, and get familiar with their usage through the test harness. Implementing these was pretty simple, the hardest part for me was getting used to the struct syntax in C. Once I made the realization that the stack can simply be monitored using the current index value, it was easy to implement the functions described in the header file. The hardest part was definitely the rpn evaluate function. I started simple here, reading up on strtok() and its usage. I used strtok() in order to break up the input string into tokens, and then I looped through the tokens, evaluating each of them. The hardest part of the evaluation was checking for invalid characters. I ended up doing this by using '' to get ASCII codes, and checking that each char in a token string was a valid one. Other than that, I had some difficulty making all of the errors work correctly, and it took some thinking through what each one meant in order to implement them correctly. Especially, I was confused about the underfill vs too few items remaining errors, because it seems to me that it would be impossible to ever hit the too few items error without getting the underfill error first. If I were to do this lab again, I would just use the ASCII codes immediately in order to check for invalid tokens. I spent quite some time trying different methods here until I landed on a good answer. Finally, I used a lot of print statements at different points to debug, and I found the gcc error codes quite useful for debugging as well.

I spent overall about 12 hours writing the code for this lab. But, since I have some experience in coding I imagine it would be a lot more difficult for other students. The main request I have would be to have more explanation of what the enumerated error codes for rpn.c are, since there is no description, just an obscure name. I also think the test harness is a bit redundant for students who debug using print statements, or other methods. I find myself having to go back and rewrite a test harness in order to make it look better, when I have already tested my functions thoroughly. Overall, I enjoyed this lab, and felt like I learned a lot while completing it. Especially, I learned more about structs in C, and their usage.