Project #: 1

Semester: Spring 2022

Name: Christopher LaFave

I. Requirements: Restate the problem specification and any detailed requirements in your own words.

- Create a command-line program that takes 4 arguments (program file, any string, integer, bigger integer) and reverses the characters in that string between the two integers.
- The reversal is done with the function reverse(char*, char*) that you need to write, using those specific parameters.
- Must detect 4 different errors
    - More or less than 4 arguments
    - Arg[2-3] are not integers
    - Last int is small that first int
    - Last int is bigger than string length

II. Design: How did you attack the problem? What choices did you make in your design, and why? Show class diagrams for more complex designs.

I started by going down the list of error checks and coding them all in. To code in most of the errors, you need to create the baseline of the code (str, frontIndex, rearIndex, etc). By the time I was done with that, all I needed to do was create *front and *rear so that I could write the reverse function.

The reverse function was relatively simple to create once I understood how it needed to be done. I made a for loop that used the beginning of frontIndex (front[0]) and the beginning of rearIndex (front[strlen(front) - strlen(rear)]). Once I found those, I could swap those two, and then push them one number towards eachother.

III. Security Analysis: State the potential security vulnerabilities of your design. How could these vulnerabilities be exploited by an adversary? What would be the impact if the vulnerability was exploited?

If somebody could get my program to point to memory that is outside of my string, it could mess with the outside memory and change things around.

IV. Implementation: Outline any interesting implementation details in your solution.

I think the way I found the segLength is pretty cool, but that's probably what you are supposed to do.

The way I made rear relative to front so that I can check if it is the right set of characters is also pretty neat.

V. Testing: Explain how you tested your program, enumerating the tests if possible. Explain why your test set was sufficient to believe that the software is working properly, i.e., what were the range of errors for which you were testing.

- Error 1:
  - Tested if it detected more or less than 4 arguments
- Error 2:
  - Tested if it accepted an int
  - Tested if it denied a string
  - Tested if it denied a string leading with an int ("1punchman")
  - Tested if it accepted negative numbers
    - Because they start with "-", it doesn't. Which could be a problem, but not in this case.
- Error 3:
  - Checked when frontIndex == rearIndex
  - Checked when frontIndex > rearIndex
- Error 4:
  - Checked when rearIndex > strlen(str)
- Reverse():
  - Did multiple tests with differently lengthed strings and segLengths to check if giving the right output

I believe that this is sufficient. I tested all the errors I needed to find, and I tested that my outputs are correct.

VI. Summary/Conclusion: Present your results. Did it work properly? Are there any limitations? NOTE: If it is an analysis-type project, this section may be significantly longer than for a simple implementation-type project.

My program seems to work properly.

There is a limitation, however. Because of the way I checked if *front and *rear were at the correct indexes, there is a chance that it chooses the wrong indexes if there is a spot in str that occurs before the frontIndex that both shares the same characters as the correct indexes and are the same distance apart.

Because that seems rare, I am choosing to not worry about it. (Also because I'm not sure how to fix it.)

VII. Lessons Learned: List any lessons learned. For example, what might you have done differently if you were going to solve this problem again?

The biggest lesson I learned in this homework assignment was how to code command-line programs. Before this assignment, I had no clue how they even worked. Now, I have quite a bit of knowledge on how to use and implement them.

Another big lesson was how to use C-strings. I'm still not fully comfortable with them, but I know have experience messing with them and will be able to figure out solutions faster.

The last big lesson from this assignment was pointers. I don't think I've ever used pointers in my code before, so traversing that confusing landscape was good experience for my next programs.