Holesum Write Up

170D WOBC: Module L Exam II (A)

CW2 Kyle Spicer

Due Date: 22 November 2022

1 Project Summary

1.1 Description

In this exercise you will write a program to identify "holes" in a data file and report the number of holes and the size of the largest hole.

A data file is an ascii text file containing an m x n matrix of zeros and ones, where holes are groups of zeros such that any zero in a group is adjacent to at least one other zero in the group. A single zero is implicitly adjacent to itself so a hole can be of size one.

Write an executable application that reads in a single ascii text file and reports the number of holes and the size of the largest hole.

2 Challenges

THOSE CRAZY END OF FILE CHARS : this was my first time encountering EOF files located in different parts of various files. This meant I could complete logic and get some tests working, but when the EOF char was located at a different point in the file, I had to go back to the beginning and complete logic to handle the EOF no matter where it was.

TAKING BREAKS : When I see my classmates making progress a lot faster than I do, it challenges me to pick up the pace. Often, I find myself spending too much time going in circles or messing up my progress because I haven't taken any breaks for hours. I need to begin watching the clock and take scheduled breaks in order to keep my mind fresh and ensure I do not make simple errors because of fatigue.

ALLOCATING MEMORY FOR 2D ARRAY : This was my first attempt at allocating and destroying memory for a 2D array. I really wanted to make sure I understood the process and how to do it properly. This portion of the program probably gave me the greatest heartache. Once I knew the array was populating properly, creating the destroy function was the next hurdle. But that proved to be pretty easy. Initially, I was trying to over free memory, then I over corrected by under-freeing the memory. Until I found the sweet spot.

COMPILING EXECUTABLE WITH ANOTHER DRIVER PROGRAM : This was my first time having to incorporate an executable I was responsible for against another driver program. I learned a lot with Makefiles and conceptualizing what the programs were supposed to do in order to work properly.

3 Successes

REFACTORING : In the beginning, I completed the program only using one of the many test files. Once I thought I was done, I quickly realized my logic only worked for certain files. I saw refactoring my code was a success, because instead of erasing code or starting over, I simply commented out everything and stepped back through my program to ensure every step was completed properly. I have grown faster in identifying issues and correcting them in my code.

PROBLEM SOLVING : I have noticed a huge improvement in myself and my ability to visualize and critically think about the problem presented to me and various way of solving it. Additionally, I am able to sketch out my thoughts in pseudo code and use print statements effectively to ensure I know what my program is doing at every step.

ORGANIZATION : With this project, I created a design plan and sketched out the entire program prior to writing a line of code. I still ran into issues and new problems that required research, however, I was always able to see the path I was on and ultimately was able to complete this project how I intended to.

GIT-TING BETTER WITH GIT : I find myself able to set up my git repositories and effectively update my documents quicker with each project. I understand the importance and actually know what I am doing and why. Also, I am able to verify what I am doing by using the website and navigating their tools.

4 Lessons Learned

ASK QUESTIONS : the instructions for this project were well written and the goals were clear. However, there were some gray areas that I needed clarification on in order to fully understand my objectives. It is essential to ask as many questions as possible.

IMPROVE MY SYSTEMATIC APPROACH TO PROBLEM SOLVING : My design phase for each project is improving. I am not simply throwing code into a program and hope I find the solution along the way. But even with a solid design plan, I still seem to go too fast and skip essential foundation parts in my program. Each step of my problem solving, when I think I have reached a checkpoint, I need to verify by completing all test cases possible to ensure each portion of my program is bullet proof.

DON'T RE-INVENT THE WHEEL : I spent a considerable amount of time attempting to work getopt() into my program after the fact. Once I realized that I only needed to provide logic to account for one additional argument, I decided to create a function to handle it. It took a few minutes, but the results came quickly and clean. I am sure there are more efficient methods, however, I was happy with the result.