Final Project

CS-499 Q1048

Melissa Caviglia O’Donnell

Southern New Hampshire University

My time as a Computer Science student has been a long journey but very rewarding. Along the way I’ve learned a lot of coding techniques, different languages and how to secure code. My student career has shaped who I’ve become as a developer and helped to focus on my professional skills. These skills include writing logical code, using object-oriented programming techniques, creating safe and secure code and using a multitude of different IDEs, testing tools, database management tools, all in various languages.

Many of my classes focused on team building exercises and taking on various roles in the Software Development Life Cycle and feeling at ease taking on these roles for various milestones. Most of my classes focused on documentation and reports as a way to communicate about projects completed as well as at milestones along the way of development. My computer science career has also introduced me a multiple database programs and structures such as SQL and NoSQL.

The first project in my portfolio is a game of Farkle that I developed within the first year of taking CS classes. It was a basic game with hundreds of lines of code that functioned and ran properly, however, needed a lot of attention to clean up. I was able to implement a lot more knowledge in the redesign of my game and was able to use multipurpose functions instead of repetitive lines of code.

The next project in my portfolio is a program written in C that strips the vowels out of user input words, groups them together according to consonant pattern and then displays each of these groups with their original word. This exercise was to provide learning on how to write linked lists and the program itself uses a combination of two linked lists to provide the words in grouped patterns. My redesign of this project was to implement a memory freeing function. Memory freeing is essential in real-life applications, so I wanted to practice my skills on this project. I chose this project, because at the time of creation and looking back at it a few terms later, this was a difficult project for me, and this was my first attempt at using pointers and linked lists. I also thought that a memory freeing function would be a great skill to demonstrate.

My last project was a Python script written to sort information found in a large database using MongoDB. I liked this project because it seemed to have real value outside of school for organizing and separating big data. The use of MongoDB is not widely used in my current career field yet, but I believe that it could be a new, innovative alternative to SQL databases that we currently use.

Overall, this portfolio showcases my skills with algorithms, coding, security, memory concerns, database management and manipulation, software design, code reviews and team collaboration skills.

**Introduction to My Artifacts**

I’ve created an informal code review [video](https://youtu.be/VquDdT4KMbg) that is available on YouTube. This will go through my chosen 3 artifacts and show an in-depth analysis of what the code looked like prior to redesign.

Software Engineering and Design: I’d like to use a dice game I created in my first year at SNHU for this category. It’s a game of Farkle written in C++. I think I’ve come a long way since I created this game and I think it’d be fun to use what I know now and improve on what I had originally. My original pseudocode is below:

Ask user to INPUT player 1 and 2 names into game

WRITE player names to score card text file

CALL text file with rules to print for players

DECLARE variables for 6 dice

Randomize dice rolling by using rand(), diving by 6, taking the remainder and adding 1

PRINT dice numbers after rolls

ASK user which dice to keep for scoring purposes

Go through each die number and ask yes or no to keep or discard and roll again

Create a loop to keep rolling and keep score unless a Farkle happens on roll 1

If a Farkle roll happens, end of turn for player and zero score is marked

WRITE player score to score card text file

Player 2’s turn will go just like player 1. Have the same loop for their rolling and scoring

WRITE player score to score card text file

CALL a sum class to keep a rolling total of both player’s scores

Once a maximum is reached, game will end and declare a winner

WRITE final score to score card text file and WRITE which player won to text file

I think I can improve upon the performance of the game features by writing more complex code. Expanding the functionality of the game and improving the code that is already in existence, although it is a rather simple game to begin with.

Algorithms and Data Structures: For this category I have a program written in C that takes a list of user input words or file input, removes the vowels and groups the consonant words by their pattern. If a word contains numbers or symbols, then it’s listed as a “bad word”. The code I developed in C contains a complex structure system and sorting system. Complex for me as this was a rather hard assignment in my opinion. Structures, pointers and linked lists are involved. I would like to keep this code in C, but I’d like to work on freeing the memory used in this program. I know at work right now, freeing used memory is important in some of our projects that are written in C, so I’d like to add that to this old assignment. Pseudocode is below:

CREATE a structure that will hold a pattern of consonants that create a linked list vertically

CREATE another structure that will hold words with that same pattern in a linked list horizontally

SCAN in user words and ensure they meet the criteria for input (no numbers or symbols)

SKIP a word that is all vowels

CALL a function that will search the vertical linked list for the consonant pattern

IF found, CALL a function that will add the entire word to the horizontal linked list

IF not found, CALL a function that will add the new pattern to the tail of the vertical LL.

CATCH any memory errors

\*ADD in a memory release function

The skills I’d like to illustrate would be enhancing the code by creating a free memory function while using more complex structures and algorithms. This has been an important requirement where I currently work and think this skill will serve me in my future career.

Databases: The artifact I’d like to choose for this category is a MongoDB database project where we tried to sort through a database of stocks and by using an aggregate pipeline, highlight the top 5 or 10 performing stocks, based on investor criteria. I think to enhance this artifact I will research what actual investors look for in stocks and create a more complex pipeline to create multiple lists of stocks performance based on certain factors. I really enjoyed working with MongoDB as opposed to MySQL, so I’d like to learn more about its functionality.

The skills I’d like expand on my MongoDB project would be more of a data mining perspective. Traversing a large database of information and sorting it by preferred criteria.

**Software Design/Engineering – Game of Farkle**

I originally created this simple game of [Farkle](https://github.com/melissaodonnell/melissaodonnell.github.io/blob/master/main.cpp) in February 2018 as part of a class learning about C++. This was one of my first-year classes and it was focused on reading and writing to text files. The game itself is a text-based game where 2 users roll 6 dice and try to arrange them to create 3-of-a-kind matches or gather 1’s and 5’s. Scoring is based on these circumstances.

I selected this artifact because it was a very rough version of a program. It was originally 559 lines of code that was all created in the main() function. There was a lot of unneeded variables that were used and a lot of redundant code. Now that it’s a year and a half later and I’ve learned more about programming, I wanted to redo this project in a more streamlined fashion. I included calling functions that had specific jobs and that eliminated a lot of the redundant code I used originally. I cut the variables by at least half and instead used arrays to hold dice values. I also have one scoring function that works when called by both players, eliminating the need for two separate but identical chunks of code that scored for each player. The redone code is now 200 lines lighter and much easier to traverse and read.

I do believe my course objective for this project was to improve on the code and its functionality and I do believe I have accomplished that with this updated version of Farkle. Updates to my outcome would be to implement more catch lines for invalid input for security reasons. Security should always be at the forefront of a programmer’s mission, and this code does not have many catches for such events.

The process was very nostalgic. I remembered things about creating this project that was so fun, but at the time, I did not have the knowledge or resources to make it amazing. Now that I know more about calling functions and arrays and for loops, I believe this code is looking much better and its functionality has been improved. I would not be embarrassed to showcase this work now. I do believe there are still challenges ahead and security improvements that could be made before my final artifact is completely polished. Overall though, I believe this project’s improvements are a great work in progress.

**Algorithms and Data Structure – No Vowels Program**

This [artifact](https://github.com/melissaodonnell/melissaodonnell.github.io/blob/master/main.cpp) was created in the fall semester of 2018 at the University of Arizona. I took one semester at a traditional university to see if I’d rather transfer there, but I realized I preferred online school. So, my first ever experience coding in C was this class and this program was developed towards the end of the semester. It was the hardest assignment I’ve ever done because linked list and pointers are still hard for me to envision and code.

I included this artifact because I think it’s a shining example of algorithms and data structures. It contains two struct features, one for a vertical linked list grouped by word patterns without their vowels. The other struct is a horizontal linked list that groups the words in the pattern by order of appearance from the input. It has a catch function for non-alpha character inputs and will print an error on the words as “Bad Word”. For this assignment, I wanted to create a memory freeing function. I know in C this is important and I’m using it a lot at work at the moment so I thought it would be a good enhancement.

I do feel like I met the objectives that I had planned by integrating a freeList() function. When using Valgrind, the function has no memory leaks and no unfree memory blocks are left behind.

This was a fun project to create in the first place, it was a challenge! So being able to incorporate a free memory function into it and enhance it in that way was great. The challenges that I faced were where to call the freeList() function in order to free all memory allocations. I did quite a bit of research to come up with the function as I wrote it. The challenge here was keeping the pointer pointed at relevant information and not freeing a node before I could point at the next one in line.

**Database – PyMongo Data Search**

This [artifact](https://github.com/melissaodonnell/melissaodonnell.github.io/blob/master/499DatabaseScript.py) was originally developed in term 19EW6, so it’s relatively new. It was part of a final project that designed an aggregate pipeline like search feature, grouping certain categories together and highlighting companies with these criteria.

I included this artifact into my portfolio because I had a fun time learning about MongoDB and using the NoSQL database. I think the skills showcased here is the ability to create a small script to sort through large databases. The script does not need to be fancy or intricate but can easily sort through large amounts of information to find something that a user wants to see.

I wanted to find a current stock database and wanted to create a python script to search through it. My submitted artifact does contain all these items. I’m pretty happy with the outcome and the ability to sort through any kind of database with a general python application. The only thing I would update, would be for security challenges. I did not put in any kind of buffer or security to check user input, which all applications should have. Since this is a private application for now, the security is not an issue.

This was a lot of fun creating. I had to research a lot more commands for PyMongo however. I used some new commands that I didn’t utilize last term for my final. I think being able to create scripts like this will be useful in my future career.