Sara Hrnciar
450 Independent Study
Dr. Gourd
Program Profiler

I initially sought to create a program which determines the likelihood of a student's work being influenced by an outside source such as chatGPT. This was a steep goal to achieve in only several months. Ultimately, I ended up with a program which takes a folder of a student's past programs and develops a "program profile," for the student. This is done by converting each program to an n by m matrix, where n is the number of lines in the current file and n is the number of predefined programming keywords. After converting each program to a matrix, they are i=nterpolated into x rows, where x is defined by the number of lines of the folder's longest program. This is done by adding new rows fitted between values of the already defined matrix. From here, each matrix's contents is scaled by 1/p, where p is the number of programs factored into the profile. This is done to ensure the program matrix's contents are sized proportionally to a single program matrix. The summed weighted matrices then give a "program profile," or a matrix which represents a student's typical program structure. From here, students' future programs are then compared to this program profile. The user is left with a similarity score from 0 and above. A higher score signals a lesser similarity.

The main tool I used to create this program profile is the NumPy library which allowed me to build matrices, find their norms, and manipulate their entries through operations like addition and division. The NumPy documentation was helpful in finding the right functions for the aforementioned operations, and defining what I should give as their parameters. I also used the SciPy library for its cubic interpolation function. SciPy offers various interpolation functions, and after consulting ChatGPT, I decided on the CubicSpline function, which fit my needs best as it provided filler rows for the given matrix. The combination of NumPy and SciPy simplified the development process and ensured the accuracy needed for my program profile.

The largest issue I've had has been figuring out the best way to approach defining the program profile. There are still questions concerning the best fit techniques for interpolation, matrix weighing, normalization, and similarity score computation. For example, there were various ways to interpolate the matrices and I chose the one which I personally felt would fit best (as described previously), but there could have been another more well-fit one. I also don't know the best norm to take the matrices, and I don't know the best way to create the similarity score. It seems that what I have right now works, but I'm sure I could use different methods to achieve more accurate results. Taking computational linear algebra was a helpful supplement to completing this project, but the topic spans much farther than what could be taught in a single semester course. I have much more to learn in order to make this tester as accurate as possible.

The result does not have a clearly defined scale of similarity. Results range from 0 and above. A result of 0 came from comparing the exact same programs, and 12 came from comparing a non-computer science text file to a program profile. This score is determined

through finding the Euclidean norm of both the student's recent program and their program profile, and taking the absolute value of their differences. Ultimately, this measures the matrice's distances from each other. An issue arises when a user is met with a similarity score of, say, 5. What does this mean? Does it hint at help from an outside source, a minor change in a student's typical program structure, or does this mean that the newest program's structure is totally different? The program score adjusts accordingly in obvious tests. The program returns a score of nearly 0 given the program profiler folder contains a text file identical to the currently tested program. It returns a similarity score of 12 when the folder contains a text file of random words. The similarity score drops closer to 0 as you add a higher quantity of identical student programs to the profiler program. Ultimately, though, the resulting number does not hold a concrete value to it and leaves interpretation up to the user.

There are endless ways to develop the program profiler. The program profiler would do well as a class, so users could then create program profile objects for numerous students. The class could contain methods to compare and add files to their program profiles. An interesting application would be to create a program profile for a collection of programs from chatGPT and to compare student's programs to chatGPT's program profile.The profiler could also have an accuracy check, which outputs how consistent in structure the programs in the program profile are. Visually, there could be a GUI where the user can drag and drop files and folders. Ultimately, there are numerous considerations I can factor in to increase the accuracy of the program, and there are a lot of additional applications that can be added to this program.