CS202 - Progress Report

Since coming up with our project idea of improving the ordering of course registration letter groups at Allegheny College we have completed a variety of tasks. These tasks included coming up with a general design for our program and writing pseudocode for the program's algorithm. Additionally, we took steps in determining how the program's input would look and how it would be read in. We also worked on writing our algorithm and the main file that will interact with the algorithm and effectively run the program. In the weeks following the creation of our final project idea, we have made great progress and faced a variety of challenges.

After determining the general design of our program, we first set off on getting input data (student data) for the algorithm. Instead of using actual student data, we agreed upon making up sample data which would include the students' name, their school year, registration group letter, and overall GPA. The data that we will be using consists of 40 students, for testing purposes, as well as the data topics previously mentioned. This data will be the input that is being read into our algorithm. The file is being read in by directing the program with a specific path that locates the designated file. Concerning the input, we had to designate each data topic with a type corresponding to what the data is. Once the file has been read in, with the correct data in each column, the file reader function is then going to go through and organize each individual student. This is done by going through each line and at every occurrence of a semicolon, the program will put that data into a tuple (with each student having their own individual tuple to contain their information) so that the information is able to be presented and utilized correctly by the program interface and algorithm.

After creating our sample input file and writing Python code to read it into the program, we began work on implementing the main file for the program which will run the algorithm and serve as the program's interface. In this file, print statements display a simple startup message

with some information about the program. Additionally, the user is also asked what input file of students they would like to use and the file reader function is then called to read this information in. This information is then fed into the algorithm, along with a sample letter group order, which is called by the interface. The sorted student information returned from the algorithm is then saved as an object, printed to the terminal, and outputted to an output file for easy readability and convenience.

After creating a simple user interface program, our group then began work on the algorithm it would be interacting with. The algorithm takes in the list containing the student information and also the letter groups. It begins by splitting the list of 8 letter groups from the previous semester into halves. It then splits these halves in half again, creating quarters or “sub-halves”. After this, it will decide to randomly swap these sub-halves around, with there being a 50% chance of a swap occurring. Then, the program will go inside the sub-halves and sort the 2 letters groups in there by calculating the average GPA of each group by going through the list of inputted students’ GPAs. With this, the letter group with the higher average GPA is moved to the first position in the sub-halve. After this, the students are then sorted based on their class year along with the new letter group order. The working algorithm then returns the sorted list of students.

During our work on this project, we have faced a variety of challenges. Most of these challenges have come in the area of implementing the algorithm, coming in the form of bugs or things just simply not working. For instance, one challenge we faced when implementing the algorithm occurred when we were trying to check if a student was in a letter group, so we could later sort them by letter group. We wrote code that made sense, but it didn’t work which confused us. To solve this challenge we printed out the inputs that were being compared and found that the student-specific letter group being read in from the input file had a space in front

of it, meaning it did not match the regular character. We fixed this issue by going into the file reader and adding support for removing spaces.

Another challenge we faced came when we had to sort the students by the proposed priority index discussed during our project proposal meeting. We realized that when we did this, it basically bypassed everything the algorithm was supposed to be as with this implementation, the students were organized by class and then by priority index instead of by letter group. This made the letter groups kind of pointless and our group felt that the class year a student is in is quite similar to this priority index idea, so we removed the priority index feature for the time being. Additionally, we have also faced the challenge of analyzing our algorithm. Since we just finished our algorithm and it has many loops to look through, we have not yet completely solved this challenge yet due to a lack of time.

The initial model/framework and pseudocode for our idea did not have to be changed. It was quite fortunate for us that we were able to implement and execute our original framework in such a manner that corresponded directly with how we originally planned to do so. Not only was this beneficial but it also made our overall progress on this project more efficient as we did not run into many hurdles that would have added time to the project. We did not have to focus our effort on thinking of new ways this algorithm could be implemented. As a result, we were able to get our most of our work done in a smaller time frame, though the algorithm did take a significant amount of time to properly implement, which allows us to have more time to go back and look over everything that we have done so far. This also gives us more time to add finishing touches and more minor features to the project.

We have made quite a bit of progress over the last few weeks. We have implemented a working algorithm and solved a variety of challenges while doing so, such as fixing the letter group inputs. Additionally, we also created a working file reader, usable sample input file,

interface, and support for outputting results to a file. With this there is still work to be done in the area of analyzing the algorithm, refactoring some of the algorithm's code, and possibly allowing for more user customization, specifically in the area of inputting the previous years' letter groups into the terminal instead of using a sample list. Overall, despite facing a variety of challenges while working on this project we have made great progress, creating a working algorithm and supporting programs.