The most challenging part of writing this program would have to be the fitness function. My understanding of the fitness function was fine, and it made sense to me on paper. The difficult part was transferring that information from paper into working code that could measure a variety of schedules (both good and bad). However, while this was the most difficult part of the program, it was also the most fun to write because it required me to be creative. It forced me to use the resources I was given (like the course schedule, a list of all the times, a list of all the schedules, etc.) and create a function that would output an accurate score. And while it was complicated, that added to the challenge of getting a working function.

All the schedules to me seem pretty good and make logical sense given the fitness function. They also seem to be optimized well. For example, each schedule has three professors each teaching at least three classes (sometimes four). This isn’t including Xu, who doesn’t have the penalty when he teaches 2 or less classes. Another thing the schedules do well is making sure CS 101 and CS 191 classes that are taught in consecutive time slots aren’t on opposite sides of campus. This is also true for professors who teach classes that are in consecutive time slots. One final thing the schedules do a good job of doing is having a primary professor teaching the class most of the time. Very rarely do you see a see a secondary choice professor teaching the class, and you never see a professor that isn’t suggested for the class that is teaching the class. However, there are some things that still confuse me about the schedule. For example, for in output2.txt file, both sections for CS 191 are taught in the same time slot. This means that there is a penalty of -0.5 is added to the fitness score. This seems like it could be easily avoided and, as a result, wouldn’t have that penalty applied.

As of right now, the program is running and working. However, that doesn’t mean there isn’t room for improvement. So, if I was given some more time, there are a couple of things I would do to improve the program. First, I would modularize the program better. There’s nothing wrong with the main program and it works fine; however, it’s just one large program with some repeated statements that could have easily been made into their own separate functions to simplify them. And not only would this simplify them, but it would also make the program easier to read and understand. The second thing I would implement is parallelization. Once again, the program runs correctly and it’s fine how it is. But parallelization would allow the program to speed up the process and allow the program to create the next generation quicker.

However, the program isn’t the only place that could use some improvements. I believe the fitness function could also use some changes. One of the first changes I would implement would be modularization. Once again, this would just allow the program to be simpler and would allow the user to read and understand the program more easily. The second thing I would change would be with the fitness function itself. As of right now, if an instructor is scheduled to teach more than 4 classes, then a penalty of -0.5 is assigned to the fitness score. However, there is a possibility for this to backfire. Say for example, we assign Professor Zein el Din to teach all the classes that are offered. Yes, the fitness score would suffer -0.5 points from Zein teaching all those classes and -1 point for having classes in the same time slot, but it would also get about 4.3 points for being the primary or secondary choice as a teacher, the potential to get 2 points for teaching the sections of CS 101 and CS 191 in consecutive time slots. Meaning, there’s the potential to have 4.8 points without even adding classroom size (which could easily add on another 3 points). Perhaps adding another, more severe penalty so that the program wouldn’t consider wasting it’s time trying out this or a similar schedule.