* + 1. What were the challenges in writing the program? Or did it seem to go smoothly from the beginning?

Designing the fitness function was complex because there were many competing requirements. One of the challenges was implementing the consecutive section for SLA101 and SLA191 activities, which require separate lists to track time slots for each section, iterating through all combinations to check for consecutive time slots, and requires couple iterations to get the implementation right. Implementing the mutate function was challenging too because it decide how to mutate each aspect of a schedule item—choosing between room, time, or facilitator randomly.

* + 1. What do you think of the schedule your program produced? Does it have anything that still looks odd or out of place?

The schedules satisfy most of the constraints in the problem, but some areas might still be less optimal. The facilitator load is unbalanced. Zeldin ends up with more activities than is ideal, while others may not assign for any activities, resulting in uneven distribution.

The schedules sometimes fail to ensure that SLA101 and SLA191 sections have optimal timing. While 2 sections of SLA 101 and SLA 191 should be more than 4 hours apart, the genetic algorithm might not always achieve the best timing arrangement due to the random nature of mutations and crossovers.

* + 1. How would you improve the program, or change the fitness function?

We can enhance the fitness function by making changes to increase penalties for frequently violated constraints. For example, if an activity is assigned to a room that is already occupied during the same time slot, the current penalty is -0.5. This penalty can be increased to -1.0 to strongly discourage overlapping activities in the same room. This will reduce the likelihood of conflicts significantly.