Problem Statement

You are given a set of $n = 15{,}000$ students in a class called CMSC 99999. Let's denote these students as $S_1, S_2, \dots, S_{15{,}000}$. Each student must find a partner in this CMSC class, and have filled out a survey indicating the following pieces of information:

- **Year**: 1 if they are a first year, 2 if they are a second year, etc.
- **Department and course number**: For the $n = 15{,}000$ dataset, the course numbers are all the same (i.e., CMSC 99999). However, for the smaller example dataset, these CMSC classes range between CMSC 15200, 12200, etc., and require filtering to in order to only pair students taking the same CMSC courses.
- **Ranked availability**: List "Morning", "Afternoon", and "Evening" in order of most preferred to least preferred.
- **Number of classes taken in the CMSC department:** Self explanatory!

Your goal is to produce an algorithm that pairs students and find a meeting time in an optimal way. Your solution should be in the form $(i, j, time)$, where i and j are indexes of students that you're pairing together and $time \in \{\text{Morning,Afternoon,Evening}\}$

What factors do you think are most important to consider? How would you construct these factors? What's the time complexity of your algorithm?

We've now been given criteria on what constitutes an "optimal" algorithm. Your goal is now to pair students with (1) similar availabilities and (2) have taken a similar number of classes in the CMSC department.

(1) **Similar availabilities**: Students would prefer to work at times that are mutually convenient. Given a solution $(i, j, time)$, you could a calculate score that takes the absolute difference between how S_i and S_j have ranked that time. For example, given a $time = "Morning"$, if S_i has ranked that as their first preference while S_j has ranked that as their third preferences, then the score would be $1 + 3 = 4$. The best score would be 2 (1+1) and the worst core would be 6 (3+3). Your goal is to maximize the sum of these scores across all 15,000 students (i.e., 7,500 pairs).

(2) **Similar number of classes taken in the CMSC department**: Students would prefer to work with people with similar skill levels as them, as measured by number of classes they've taken in the CMSC department. Similar, you calculate a score that's the absolute difference between number of classes that two students have taken. That is given a solution $(i, j, time)$, if S_i has taken 2 CMSC classes and S_j has taken 5 CMSC classes, the their score would be $|2 - 5| = 3$. Your goal would then be to minimize the sum of these scores across all 15,000 students (i.e., 7,500 pairs).

(3) **No lonely students**: Since we provide a dataset with even number of students, no one should be left alone! There would be huge penalty if you leave someone alone and you should pair togher as many people as possible.

Moreover, your goal is to make your algorithm as efficient in runtime as possible. What's the time complexity of your algorithm