One question of your choice (either #1 or #2 or #3) is due by **5:00PM Tuesday, April 9** and will be graded on completion only (1 point). Your full answers to all questions are due by **5:00PM Friday, April 12**. Each solution will be graded for correctness and clarity (4 points per question). Read the course information page/syllabus for further info about this 4-point scale.

1. You've just been elected to the Carleton Student Association (CSA) and are working to put together small working groups across all class years that will make recommendations for the initiatives the CSA will pursue in the future. Each small working group will be led by one *CSA representative* and will also contain 4 *student volunteers* - one first-year, one sophomore, one junior, and one senior. You aren't sure yet how many student volunteers you will get, so for now you are planning to have n people from each of these 5 groups (CSA, first year, sophomore, junior, and senior). In summary, you are working to coordinate $5n$ total people into n total small groups.

   You want to create groups that are as happy as possible with their group mates. Your plan is to ask all of the participants to provide their ordered preferences over all the other $5n - 1$ people, from which your algorithm will derive the working groups.

   You decide to define a *group instability* as follows: two student volunteers x and y form a group instability if they are from the same class year and they both prefer each of the members of the other's group to the corresponding members of their own.

   For example, suppose $x \in G_1$ and $y \in G_2$ are both juniors. Then x and y form a group instability if x prefers the first-year in G_2 to the first-year in G_1 AND y prefers the first-year in G_1 to the first-year in G_2, and similarly for the sophomores, seniors, and CSA representatives in G_1 and G_2.

   Notice, that this definition only allows a group-instability to be formed by two student-volunteers from the same class year. Also, it's important to note that under this definition two CSA representatives can't form a group-instability.

   Your job is to design an efficient algorithm to create a set of groups such that there are no group instabilities. You will also need to provide a convincing argument that your algorithm does just that and an analysis of its runtime in terms of n.

   Hint: to get started, think about the case where you have only CSA representatives and seniors and solve this smaller problem first. Now can you find a way to include the juniors? Then the sophomores? Then the first-years? Also, notice that this definition of stability is quite different from that in a standard stable matching – among other things, here an instability is formed by two people of the same year, whereas in the stable-matching setting an instability is formed by two people of different types.

2. Congratulations, you've been hired by the TV network ABC after graduating from Carleton! You have been put in charge of using the Gale-Shapley algorithm to match up n professional dancers to n celebrities for the TV show Dancing with the Stars. As you are working on your implementation you notice that there are some celebrities that are consistently highly ranked by the professionals. You are starting to wonder how this might impact the matches output by the algorithm.

Specifically, you start to consider the following scenario. A subset of k of the n celebrities are more popular than the other $n - k$ celebrities; they're so popular that every professional has these same k celebrities as their top k choices (though they may not agree on the ordering of the k celebrities). You want to figure out what will happen to these k celebrities if they do the asking (proposing) for the G-S algorithm. You conjecture that these celebrities will always end up with one of their top k choices of professionals.

(a) Often, doing small examples before trying to prove something can help convince you that what you hope to prove is actually true, and can give you additional context for when you start your proof. With this in mind, create a small example for the scenario described above where $n = 5$ and $k = 3$ and run Gale-Shapley to see if what happens matches your conjecture. Include your starting preference lists, the sequence of proposals, and the final matches in your write-up. (You may actually want to try this more times with different n and k, but I only ask you to include the one specified example in your write-up).

(b) Let c be one of the celebrities who is ranked in the top k choices by all of the professionals. Prove that if the Gale-Shapley algorithm is run with the celebrities doing the asking (proposing), then celebrity c will always end up with one of their top k preferred professionals.

Hint: There are many ways that you might go about proving this. I found proof by contradiction to be a good technique to use here. If you don't remember how that works, and you aren't sure where to start - I'd suggest reviewing that proof technique.

3. In this exercise, you are going to consider a particular instance of the stable matching problem where you are matching n piano teachers to n students, and all n piano students have identical preferences over the piano teachers.

(a) Write out a small example with $n = 4$ and preference lists that have the properties described above. Run through the Gale-Shapley algorithm with the students doing the asking. Include your starting preference lists, the sequence of proposals (so you can count exactly how many proposals occur), and the final matches in your write-up.

(b) Why might it be useful to know the lower-bound on the number of proposals that happen during the execution of the Gale-Shapley algorithm? (∼1-2 sentences should suffice.)

(c) Write a convincing argument that the Gale-Shapley algorithm requires $\Omega(n^2)$ proposals on any instance with the properties described above.