CS 461
Fall 2017
Due night of Sunday, Feb 25

This program will use simulated annealing to attempt to optimize a schedule.

The Computer Science department at Wassamatta U requires all CS 101 students to take a lab section as part of the course. There are 20 seats in the computer lab, and about 100 students in the course. Thus, at least 5 lab sections are needed. In the past, Wassamatta U has offered 7 lab sections, so there were empty seats in every lab and everyone could take whichever section they wanted. But with budget cutbacks, the school can no longer afford the 'extra' sections and is offering the minimum number possible—5 sections. Of course, students have different preferences as to which lab they would like to attend, but some times are more popular than others. The school used a first-come first-enrolled method last semester, but the most popular time filled within an hour of enrollment opening and a brawl broke out among computer science students in the administrative center. (It was a geek fight of epic proportions, including such insults as "Go dereference a null pointer!" and "Your mama's so fat she has to be addressed in 128 bit!")

Clearly, we wish to avoid such an outcome again.

So this semester, students are asked to rank their preferences. There are 5 lab sections, conveniently labeled A, B, C, D, and E. To avoid bias, each student will be assigned a unique number from 1 to 100. Students were instructed to list all 5 labs in order of preference. Some students have listed only a portion of the labs, with X indicating none of the other labs are suitable. These students may not match for a lab. In all cases, once an X is listed, the rest of the student's listings are also X. Each letter other than X appears only once. You are given a file containing the student's rankings, and need to produce an assignment of students to lab sections.

You will use *simulated annealing* to address this problem. The metric you will try to minimize is the square of the student's preference ranking for a section; that is, someone in their first choice has a score of 1, someone in their 5th choice has a score of 25, and lower scores are better. (Squares are used because the department chair has decided that having 2 students each in their second choice is better than 1 in their first choice and 1 in their third; using just the preferences, we would consider 2+2 == 1+3 and so these would be equal. Using squares, 4+4 < 1 + 9 and so we'd prefer the more egalitarian ranking.

Start with a fairly high T parameter (analogue of temperature), such that most potential changes are accepted. Then continue making changes according to the usual algorithm; every 10N (1000) modifications or 100N (10,000) attempts, reduce T by 5%. For this problem, the probability of taking a step that harms (increases) the fitness score is

$$P(Change) = e^{-(S_2 - S_1)/T}$$

where S_2 is the proposed new state, and S_1 is the unmodified 'old' state. So if the new state has a lower (better) fitness score than the old one, the exponent will be positive (> 0) and the probability will be 1. If the new state has a lower fitness than the old one, use the exponential formula and the random-number generator to determine if the change should be made or not.

You are given 2 input files, of similar size. Note that because you are using simulated annealing, you may not get the same matching every time, and therefore not the same score.

This is a variant on the *Hospitals and Residents* problem, which is itself a special case of the *Marriage problem.* At the conclusion of medical school, new doctors interview at various residency programs, and rank their preferences for where they would like to go. Residency programs rank the people they have interviewed based on who they would like to accept. This data is then fed to a matching algorithm, and the new doctors are legally obligated to go to the residency program they are matched with.

Read up a little on the hospitals & residents problem and write a short report (a couple of pages, double-spaced) discussing the differences between your program and the algorithm used in those cases. Submit it along with your source code. Your program can be in C++, Java, or Python.