Introduction to Algorithms	Problem Set 1 (3 problems)
CS 4820 Fall 2019	Due 11:59pm Thursday, September 12

There are three questions on this homework. A coding problem that was released last week, and two written questions.

Your homework submissions for written questions need to be typeset (hand-drawn figures are OK). See the course web page for suggestions on typing formulas.

The solution to each question needs to be uploaded to CMS as a separate pdf file. To help provide anonymity in your grading, do not write your name on the homework (CMS will know it's your submission).

Collaboration is encouraged while solving the problems, but

1. list the names of those with whom you collaborated you (as a separate file submitted on CMS);
2. you must write up the solutions in your own words;
3. you must write your own code.

Remember that when a problem asks you to design an algorithm, you must also prove the algorithm's correctness and analyze its running time. The running time must be bounded by a polynomial function of the input size.

(1) Implementing Gale-Shapley. In this problem we consider the classical Gale-Shapley algorithm. Given n proposers (classically men) and n respondents (classically women), and a preference list for each proposer and respondent of all members of the opposite sex, give a implementation of the Gale-Shapley stable matching algorithm with men proposing that runs in $O(n^2)$ time, as explained on page 46 of the textbook. Implement the algorithm in Java. The only libraries you are allowed to `import` are the ones in `java.util.*`, and ones dealing with reading and writing inputs, such as `java.io.Reader`, `java.io.Writer`, etc. We recommend using BufferedReader and BufferedWriter for reading/writing standard input/output.

Warning: be aware that the running time of calling a method of a built-in Java class is usually not constant time, and take this into account when you think about the overall running time of your code. For instance, if you used a LinkedList, and use the `indexOf` method, this will take time linear in the number of elements in the list.

We have set up an **online autograder** at `https://cs4820.cs.cornell.edu/`. You can upload solutions as many times as you like before the homework deadline. **You need to have the main method of your code named Main, must not be "public", must not be part of a package**. When you upload a submission, the autograder will automatically compile and run it on a number of public testcases that we have prepared for you, checking the result for correctness and outputting any problems that may occur. You should use this facility to verify that you have interpreted the assignment specification correctly. After the deadline elapses, your submission will be tested against a new set of *private* test cases, which your grade for the assignment will be based on.

Input / output formatting and requirements

Your algorithm is to read data from `stdin` in the following format:

- First line has one integer number, $1 \leq n \leq 1000$. Proposers and respondents are labeled with numbers $0, 1, \ldots, n-1$.
- In each of the next n lines, we are providing the preference list of a proposer. The ith line is the preference list of the ith proposer (the first respondent in the list is the most preferred and the last respondent is the least preferred).
- In each of the next n lines, we are providing the preference list of a respondent. The ith line is the preference list of the ith respondent (the first proposer in the list is the most preferred and the last proposer is the least preferred).

Your algorithm should output data to `stdout` in the following format:

- 2 lines, the first line containing number r_1 and the second line containing number r_2, described below:
  - r_1 should be the number of the proposer that respondent 0 is matched to in your stable matching.
  - r_2 should be calculated as follows: after implementing Gale-Shapley, modify your implementation by changing the code so that the first proposal by man m_0 to the woman w first in his list is rejected even if his favorite partner/woman W is unmatched or would prefer m_0 to her current partner. (All other woman behave the same as in the classical algorithm, and m_0's favorite partner w also behaves the same at all times expect when m_0 proposes. This changes (untruthful) behavior of w can result in a few different outcomes:
    1. woman w may not get matched at all.
    2. woman w may be part of an instability.
    3. woman w may be matched to the same partner or a better partner than in the matching resulting from the standard algorithm.

    r_2 should be equal to the number of the case that occurred.

For example, when the input is:

```
2
0 1
1 0
1 0
0 1
```

There are two men and two women, and in the classical Gale-Shapley algorithm runs as follows:

1. man m_0 proposes to woman w_0,
2. man m_1 proposes to woman w_1,

and a stable matching is formed consisting of $(m_0, w_0), (m_1, w_1)$. So the answer for part (a) is 0.

For part (b) the first proposal by man m_0 is rejected, so this algorithm proceeds as follows

1. men m_0 proposes to women w_0, and gets rejected (due to the changed rule)
2. men m_0 proposes to women w_1,
3. men m_1 proposes to women w_1 and gets rejected as w_1 prefers m_1
4. men m_1 proposes to women w_0

This results in the matching $(m_0, w_1), (m_1, w_0)$, which is stable, and women w_0 has an improved partner, so the answer to part (b) is 3. Thus, your output for this test case should look like:

```
0
3
```

In contrast, if the input was

```
2
0 1
1 0
0 1
1 0
```

the answer to part (a) is the same, but the result for part (b) is that w (which is w_0 again) will remain unmatched, so the answer to part (b) is 1, and thus your output for this test case should look like:

```
0
1
```

(2) Stable matching variants. Recall that in the Gale-Shapley algorithm all proposals are made only on one side (classically the men's side), and that the results favors the proposer's side (as showed in class on Wednesday, and a;sp see claim (1.7) on page 10 of the textbook). A more fair way to try to find a stable matching is to allow both sides to make proposals. Here is what such an algorithm may look like:

```
maintain a 2n long array CURRENT.
   For each person i (man or woman) maintain a pointer: CURRENT[i] which is the current
       matched partner of i (nill if i is not matched)
maintain two n-by-n tables REJECTED. For a man M and woman W, REJECTED[M,W]=REJECTED[W,M]
       are both set to 1 if either M rejected W or W rejected M, and 0 otherwise,
While there is a person i (man or woman), and a possible partner j such that
   REJECTED[i,j]=0, and i is currently unmatched or i prefers j to his/her
   current partner CURRENT[i] then
       i makes a proposal to the highest ranked j of this form
       if j is unmatched or prefers i to his/her current partner, then
           j rejects CURRENT[j] (if not nil) and tentatively matches with i, that is:
               if CURRENT[j]=x not nil, then
                   update CURRENT[x]=nil and REJECTED[j,x]=REJECTED[x,j]=1
               if CURRENT[i]=y not nil, then
                   update CURRENT[y]=nil and REJECTED[i,y]=REJECTED[y,i]=1
               set CURRENT[j]=i, CURRENT[i]=j,
       else
           j rejects i, that is
               set REJECTED[i,j]=REJECTED[j,i]=1
endwhile
Return current tentative matching
```

in class we showed that the Gale-Shapley algorithm has the following useful properties: there were only at most n^2 proposals, and the algorithm was guaranteed to output a stable perfect matching. For each of these claims either prove that the claim remains true in the new version also, or show a counter example that this claim can fail (that is, an input to the stable matching problem along with a way to run the algorithm above that makes the claim fail). Concretely, answer the following questions

(a) (2 points) Is the new symmetric algorithm above guaranteed to end after at most $O(n^2)$ proposals (assuming that there are n men and n women).
(b) (2 points) is the algorithm guaranteed to output a perfect matching?
(c) (3 points) If the resulting matching is perfect, is it guaranteed to be also stable?

(d) (3 points) Suppose you run the above algorithm but give priority to one side (say the man), that is whenever possible the person i selected is a man, and only let women propose once no man can propose. Answer questions (a-c) for this version of the algorithm also.

(3) Scheduling check-ups. (10 points) You are helping a supervisor in a sport facility. With the year just starting they are hiring a lot of new helpers in the gym for various activities. Each helper works for one shift each day. The supervisor would like to visit the gym a few times the first day, and make sure she meets all new helpers. This question is asking you to give an efficient algorithm for this problem. To be more formal, assume there are n new helpers, and helper i has a shift for the time interval $[s_i, f_i]$. If the supervisor visits at time t, she meets all helpers whose interval satisfies $s_i \leq t \leq f_i$.

Give a polynomial time algorithm that finds a set of visit times T with $|T|$ as small as possible so that the supervisor meets all helpers. For full credit your algorithm should run in $O(n \log n)$ time.