**Introduction to Algorithms** | **Problem Set 1**
**CS 4820, Spring 2019** | **Due 11:59pm Thursday, January 31**

**Hand in your solutions electronically using CMS. Each solution should be submitted as a separate file. Collaboration is encouraged while solving the problems, but:**

1. **list the names of those with whom you collaborated;**
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)** *(5 points)* For each positive integer $n$, let $t_n$ denote the number of distinct ways to cover a rectangular $2 \times n$ grid with non-overlapping dominoes. What is the value of $t_n$? Prove the correctness of your answer using mathematical induction.

**Hint:** You are allowed to use `https://oeis.org/`. This may help if you know how to calculate the elements of the sequence $t_1, t_2, t_3, \ldots$ but you don't know what terms to use to describe the sequence.

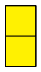
Figure 1: $t_1 = 1$

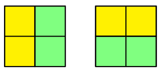
Figure 2: $t_2 = 2$

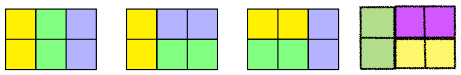
Figure 3: $t_3 = 3$

**(2)** *(10 points)* Suppose we are given an instance of the stable matching problem, consisting of a set of $n$ applicants $\{x_1, \ldots, x_n\}$ and a set of $n$ employers $\{y_1, \ldots, y_n\}$, together with a list for each entity (applicant or employer) that ranks the entities of the opposite type from best to worst. This exercise concerns algorithms to solve the following problem: *decide whether there exists a stable perfect matching in which $x_n$ is matched to $y_n$.*

**(2a)** A simple algorithm for this problem is the following: remove $x_n$ from every employer's preference list, and remove $y_n$ from every applicant's preference list. Run the Gale-Shapley algorithm (say, with employers proposing) to find a stable perfect matching, $M$, of the applicant set $\{x_1, \ldots, x_{n-1}\}$ and employer set $\{y_1, \ldots, y_{n-1}\}$. If $M \cup \{(y_n, x_n)\}$ is a stable perfect matching of the original $2n$ entities (with their original unmodified preference lists) then answer "yes"; otherwise, answer "no". Give an explicit input instance on which this algorithm outputs the wrong answer.

**(2b)** Design a polynomial-time algorithm to decide whether there exists a stable perfect matching in which $x_n$ is matched to $y_n$. Prove that your algorithm always outputs the correct answer and analyze its running time.

**Hint:** The solved exercises at the end of Chapter 1 in the textbook may provide a useful subroutine for your algorithm.

**(3)** *(10 points)* In this problem you are asked to implement the Gale-Shapley stable matching algorithm **with employers proposing to applicants**. Your implementation should run in $O(n^2)$ time, as explained on page 46 of the book.

Implement the algorithm in Java using the environment provided — use the framework code (Framework.java) we provide on CMS, to read the input and write the output in a specific form (this makes it easy for us to test your algorithm). The only thing you need to implement is the algorithm, and you are restricted to implement this between the lines `//YOUR CODE STARTS HERE` and `//YOUR CODE ENDS HERE`. This is to make sure you can only use classes from `java.util` (imported at the start of the file).

**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 use a LinkedList, and use the indexOf method, this will take time linear in the number of elements in the list.**

You can test your code with the test cases provided on CMS. Framework.java takes two command line arguments, the first is the name of the input file, and the second is the name of the output file. The input file should be in the same folder in which your compiled java code is. After you compile and run your code, the output file will also be in the same folder. In order to test your code with the provided test cases, copy the test cases in the folder in which you have compiled your code, and set the name of the input file to be the name of one of the sample inputs (`Test`*i*`in.txt` for $i = 0, 1, \dots, 4$). Each of the provided sample outputs (`Test`*i*`out.txt` for $i = 0, 1, \dots, 4$) are the output of the Gale-Shapley algorithm when employers propose to applicants for the corresponding sample test case. The first four test examples are small; you can use these to help test the correctness of your code by running the algorithm, and checking if your code generates the same output as the one we gave you. (Note that the resulting matching does not depend on the order of proposals made.) The larger instances are useful to help test the running time. The last two have $n = 1000$ and $n = 2000$. The running time of your code should increase quadratically, not cubically, as the input gets bigger. We expect that even the $n = 2000$ instance should take less than 1-2 seconds to run if your code is $O(n^2)$.

The format of the input file is the following:

- First line has one number, $n$. Both applicants and employers are labeled with numbers $0, \dots, n-1$.
- In each of the next $n$ lines, we are providing the preference list of an applicant. The $i$th line is the preference list of the $i$th applicant (the first employer in the list is the most preferred and the last employer is the least preferred).
- In each of the next $n$ lines, we are providing the preference list of an employer. The $i$th line is the preference list of the $i$th employer (the first applicant in the list is the most preferred and the last applicant is the least preferred).

Each line of the output file corresponds to an employer and an applicant that are matched by the algorithm. The employer is listed first and the applicant second.

The code reads in the input, and stores this in two $n \times n$ matrices: APrefs and EPrefs, where row $i$ of the APrefs matrix lists the choices of applicant $i$ in order (first employer is most preferred by applicant $i$), and similarly, row $i$ of the EPrefs lists the choices of employer $i$ in order. Your code needs to output a stable matching by putting each matched pair in the MatchedPairsList and needs to run in $O(n^2)$ time.

We use Java 8 for compiling and testing your program.

**Remember: The problem asks you to implement the *employer-proposing* version of the Gale-Shapley algorithm.**