COMP 482 Project 1: Checking Stability
Due: Wednesday October 5 at 2355
Points: 30 points possible

Overview: Your program will be given an instance of StableMarriage and a matching. You will check whether the matching has an instability. If it does you will modify the matching by pairing the couple forming the instability and pairing the partners of the couple forming the instability and then repeat this process until either the resulting matching is stable or you loop 100 times. In either case, you will output the number of times you modified the matching and output the resulting matching. Note that if given a stable matching you will output 0 and and the matching given.

Details: The input will come from a file called input.txt which will be placed in the same directory as your java file. The first line of the file will have a single integer value N which will be the number of men (or women since the number of men equals the number of women). The next N lines will be the whitespace separated preference lists of the N men (ie each of the next lines will be a permutation of 1, 2, ..., N). The next N lines will be the whitespace separated preference lists of the N women. The next N lines will be a whitespace separated matching. See the sample input below for examples.

While the matching is unstable and less than 100 changes have been made, your program will check whether the given matching is stable. If it has an instability involving M_i and W_j then you will change the matching by pairing M_i and W_j AND pairing M_i's partner with W_j's partner. Once 100 changes have been made or the matching is stable, your program will output the number of changes and the final matching. For some input solutions may vary because of which order you search for instabilities. For the examples below, I check for instabilities in the order M_1 with $W_1, W_2, \ldots, W_N$ and then M_2 with $W_1, W_2, \ldots, W_N$, ..., M_N with $W_1, W_2, \ldots W_N$.

Checking whether a matching is stable can be done the same way as in class (ie for every pair (i, j) you check if M_i prefers W_j to his partner and whether W_j perfers M_i to her partner).

You can discuss ideas for the algorithm to be used with anyone and consult any source (books, internet, etc). However, for this project, you are expected to write the code on your own with limited or no assistance from the professor, no assistance from others, and limited or no assistance from other sources (books, internet, etc). To clarify, you can seek assistance in understanding the task and how it can be solved, but "your code" should be written by you: not written by others, not copied from others, not copied from books/internet.

Picky, but required specifications: Your project must:

- be submitted via canvas.
- consist of 1 or more dot-java files (no class files, zip files, input files or other files should be submitted). Each file must have your name and which project you are submitting as comments on the first 2 lines.
- not be placed into any package (for the java pedants, it must be in the default package).
- have one file called Project1.java.
- compile using the command 'javac Project1.java'.
- run using the command 'java Project1',
- accept input from a file called input.txt in the same directory as the java file(s) formatted precisely as described above.
- be submitted on time (early and multiple times is fine).

If your project fails any of the above, you will receive a zero (recall that each of you may replace 1 and only 1 project that receives a zero this semester). If your project meets the requirements above then it will be graded on whether it:

- is designed and formatted reasonably (correct indentation, no excessively long lines, no excessively long methods, has useful method/variable names, etc) and
- accomplishes the goal of the project. In other words, the output should be the correct answer, computed in a valid way, formatted correctly.

Sample execution: If input.txt contains

```
3
1 2 3
2 3 1
3 1 2
3 2 1
1 3 2
2 1 3
1 2
2 3
3 1
```

then the output should be

```
0
1 2
2 3
3 1
```

since the matching given is stable.

If input.txt contains

```
3
1 2 3
2 3 1
3 1 2
3 2 1
1 3 2
2 1 3
1 3
2 1
3 2
```

then the output could be

```
2
1 2
2 3
3 1
```

because

- the initial matching has the instability $M_1 : W_2$ which changes the matching to $(M_1, W_2), (M_2, W_1), (M_3, W_3)$,
- the new matching has the instability $M_2 : W_3$ which changes the matching to $(M_1, W_2), (M_2, W_3), (M_3, W_1)$, and
- the new matching is stable.

If input.txt contains

```
5
1 2 3 4 5
2 3 1 5 4
4 5 3 1 2
5 3 2 1 4
4 1 3 2 5
2 1 3 4 5
1 2 3 4 5
3 1 4 2 5
2 4 5 1 3
3 5 1 4 2
1 2
2 1
3 3
4 4
5 5
```

then the output could be

```
5
1 1
2 2
3 5
4 3
5 4
```

because

- the original matching has instability $M_3 : W_5$ which changes the matching to $(M_1, W_2), (M_2, W_1), (M_3, W_5), (M_4, W_4), (M_5, W_3)$
- the new matching has instability $M_2 : W_3$ which changes the matching to $(M_1, W_2), (M_2, W_3), (M_3, W_5), (M_4, W_4), (M_5, W_1)$
- the new matching has instability $M_1 : W_1$ which changes the matching to $(M_1, W_1), (M_2, W_3), (M_3, W_5), (M_4, W_4), (M_5, W_2)$
- the new matching has instability $M_2 : W_2$ which changes the matching to $(M_1, W_1), (M_2, W_2), (M_3, W_5), (M_4, W_4), (M_5, W_3)$
- the new matching has instability $M_4 : W_3$ which changes the matching to $(M_1, W_1), (M_2, W_2), (M_3, W_5), (M_4, W_3), (M_5, W_4)$ and
- the new matching is stable.

Stray Thoughts:
I suggest you finish and submit your project at least several days in advance. This way you have time and opportunity to ask any last questions and verify that what you upload satisfies the requirements. There is nothing wrong with working on the project for a day and uploading your code, working for another day and uploading your improved code, ..., working for another day and uploading your final version. In fact there are advantages: you have a fairly reliable place that keeps your versions and even if you get busy at the last moment you have still uploaded your best version.
Your project should be written and understood by you. Helping or receiving help from others to figure out what is allowed/required is fine, but copying code is not. Significant shared source code indicates that you either did not write/understand what you submitted or you provided code to another (allowing them to submit code they did not write/understand). Both are academic dishonesty.