COMP 610 Project 1

Idea: An instance of the StableMatching problem can have one or multiple solutions (stable matches). For a given instance of StableMatching your program will find

1. the maximum number of men who can have their first choice in a solution,
2. the maximum number of women who can have their first choice in a solution, and
3. the maximum number of people who can have their first choice in a solution.

Input Format: The input file will be called input1.txt and be in the same directory as the java and class files. The format of input1.txt will be a standard text file. The first line will be the value of n (the number of men - or women). The next n lines will be the preferences of the n men. The next n lines will be the preferences of the n women.

Output: Your output will be 3 integers: the maximum number of men who get their first choice in a stable matching, the maximum number of women who can get their first choice in a stable matching. The maximum number of people who can get their first choice in a stable matching.

Examples: If input2.txt contained

```
2
1 2
2 1
1 2
2 1
```

then the output woud be

```
2
2
4
```

If input2.txt contained

```
2
1 2
2 1
2 1
1 2
```

then the output woud be

```
2
2
2
```

If input2.txt contained

```
2
1 2
2 1
1 2
1 2
```

then the output would be

```
2
1
3
```

Details: The program must be written in Java. The program must compile with the command javac *.java and run with the command java Project1. The program should be reasonably commented, indented, structured. The program should be submitted by placing all files in a directory named after you, zipping this directory and submitting via canvas (ie if the professor was submitting then all files would be placed in a directory called JohnNoga, this directory would be zipped, and uploaded in canvas). Failure to follow these directions will result in serious reductions in your score.

How: I'm open to any method which computes the correct answers (even brute force). There is an $O(n^2)$ way to find the first 2 ouput values. I don't know of an efficient way to find the third, but I haven't though about it much. So there might be an efficient way to do it. There is an obvious brute force method which is $O(n^2 n!)$. If you come up with a significantly faster method, I'd be curious.