Computer Science 1 – Program 3
Match-Making (RECURSION)
Assigned: 2/9/11
Due: 9/23/11 (Wednesday) at 11:55pm (WebCourses time)

The Problem

Jonathan has noticed that there are many match-making websites and event planners out there, but none of them seem to have very much success, on average. He is confident that he has an idea that is sure to beat all the others out there. Most of the websites out there ask clients all sorts of information and then produce a match score between pairs of people based on the answers to these questions. From there, anyone is free to email anyone else and can view the corresponding match score.

Of course, without meeting someone, you can never really predict if you might like them. Thus, Jonathan has decided that if he is to make a match-making business, he must get people to meet in person. An example of such an event is speed-dating, where each of the participants has several mini-dates. At the end of the night, participants go to a website and for each person they met, they simply state whether or not they are interested in that person. If there is a "match", each person receives the other's contact information. We may assume that any speed-dating session has an identical number of men and women.

What's particularly troubling to Jonathan about the speed-dating scenarios is that some people may get many matches while others get none. This does not appeal to his egalitarian principles. He would prefer that after a round of speed-dating, each person gets matched with exactly one person of the opposite sex. Since Jonathan is a numbers guy, he would prefer that each person rate each person they have a mini-date with using a number from 1-10, represent how much they like that person. Using this data, Jonathan is sure he can match everyone up in a way that maximizes the total group's "likeability" quotient.

A matching is simply a listing of each female in the group paired with an individual male.

The "likeability" quotient of a matching is as follows:

For each couple in the matching, look at how much the man likes the woman and the woman likes the man. (Both of these will be numbers from 1 to 10.) Record the minimum of the two numbers.

Add each of these numbers for each couple in the matching to determine the likeability quotient of the matching.

Since Jonathan is too busy heading the marketing division of the company, he has asked you to write the software that will calculate the matching that leads to a group's maximum "likeability" quotient. Each group will be relatively small (no more than 10 men and 10 women). Thus, you can try all possible matchings (no more than 10!) and determine the one that leads to the greatest "likeability" quotient.

If more than one matching leads to the same maximal "likeability" quotient, your goal will be to minimize the difference in likeability between all of the pairs. Just go through each couple in the matching and add up the differences in how much they like each other. (If his score for her is 5 and her score for him is 2, then the difference is 3.) In case of ties, the matching you want to pick is the one that minimizes this sum.

The reasoning behind these metrics is fairly clear:

If one person likes the other person more, then as a pair, their liking for each other is only as strong as the person who likes the other person less.

Secondly, it's generally bad to have a relationship where one person likes the other much, much more, since it creates an inequality. Thus, minimizing this inequality ought to be a way to break ties between competitive matchings.

Let's go through a simple example with 3 men and 3 women:

Let the men's names be Adam, Bob and Carl, and the women's names be Diana, Ellen and Fran.

The following chart shows how much the men like the women:

	Diana	Ellen	Fran
Adam	4	8	7
Bob	6	7	5
Carl	5	9	6

This chart shows how much the women like the men:

	Adam	Bob	Carl
Diana	7	6	8
Ellen	6	5	9
Fran	4	7	3

Now, let's look at the six possible matchings and their scores

M1	S	M2	S	M3	S
Adam+Diana	4	Adam+Diana	4	Adam+Ellen	6
Bob+Ellen	5	Bob+Fran	5	Bob+Diana	6
Carl+Fran	3	Carl+Ellen	9	Carl+Fran	3
Total	12		18		15

M4	S	M5	S	M6	S
Adam+Ellen	6	Adam+Fran	4	Adam+Fran	4
Bob+Fran	5	Bob+Diana	6	Bob+Ellen	5
Carl+Diana	5	Carl+Ellen	9	Carl+Diana	5
Total	16		19		14

In this situation, the best matching is the fifth one placing Adam with Fran, Bob with Diana, and Carl with Ellen, for a "likeability" quotient of 19.

Had there been a tie, we would have broken it by looking at the differences in the matching. For matching #5, the difference between Adam and Fran is 3, Bob and Diana is 0, and Carl and Ellen is 0, for a total difference of 3+0+0 = 3.

The input will be such that there will always be a unique matching that is optimal based on these two criteria.

Input File Specification (input.txt)

You will read in input from a file, "input.txt". The name MUST BE "input.txt". Have this AUTOMATED. Do not ask the user to enter "input.txt". You should read in this automatically. (This will expedite the grading process.)

The first line of the file will contain a single positive integer, *n*, representing the number of speed-dating events. Each of the *n* test cases will follow. The first line of each test case will contain a single positive integer, *c (1 < c < 11)*, representing the number of couples in the test case. The second line of each test case will contain *c* alphabetic strings (19 or fewer characters) separated by spaces representing the names of the men, from man #0 to man *#(c-1)*. The third line of each test case will contain *c* alphabetic strings (19 or fewer characters) separated by spaces representing the names of the women, from woman #0 to woman *#(c-1)*. The next *c* lines will contain the men's ratings of the women. The first line will contain the ratings of man #0, the second line, the ratings of man #1, etc. Each of these ratings will be positive integers from 1 to 10, inclusive, separated by spaces. The first rating on each of these lines is for woman #0, the second rating is for woman #1, etc. the last *c* lines of each test case will contain the women's ratings of the men, with the first line being woman #0's ratings, the second line being woman #1's ratings, etc. Similarly, on each of these lines, the first integer will be the rating of man #0, the second integer will be the rating of man #1, etc, and all of these ratings will be separated by spaces as well.

Output Format

*****NOTE***: You should generate your output to a FILE that you will call "out.txt".**

For each test case, output the following header:

```
Matching #k: Maximum Score = X.
```

where k ($1 \leq k \leq n$) represents the speed dating session, and X represents the maximum "likeability" quotient for the group.

Skip a blank line, and follow with each pair in the matching, the man's name followed by the woman's name, separated by a space. The ordering of the matching should be in the same ordering the men were given in the input file.

After each test case, skip two blank lines.

*****WARNING*****

Your program MUST adhere to this EXACT format (spacing capitalization, use of dollar signs, periods, punctuation, etc). The graders will use very large input files, resulting in very large output files. As such, the graders will use text comparison programs to compare your output to the correct output. If, for example, you have two spaces between in the output when there should be only one space, this will show up as an error even through you may have the program correct. You WILL get points off if this is the case, which is why this is being explained in detail. Minimum deduction will be 10% of the grade, as the graders will be forced to go to text editing of your program in order to give you an accurate grade. So as an example, here is the first format shown above (if the instruction was an addition):

Again, your output MUST ADHERE EXACTLY to the sample output shown below.

Sample Input File

```
2
3
Adam Bob Carl
Diana Ellen Fran
4 8 7
6 7 5
5 9 6
7 6 8
6 5 9
4 7 3
2
Brad Tom
Katie Angelina
5 8
9 7
4 8
7 6
```

Corresponding Output (saved to file called "out.txt")
PAY ATTENTION TO THE EXACT FORMAT (SPACING, PUNCUATION, ETC.)

```
Matching #1: Maximum Score = 19.

Adam Fran
Bob Diana
Carl Ellen


Matching #2: Maximum Score = 15.

Brad Angelina
Tom Katie
```

NOTE:

Your program MUST use recursion. SPECIFICALLY, your program MUST use the recursive PERMUTATION algorithm that we discussed in class. You will use this to get all the different permutations of matches. That is the purpose of this assignment. You will get no more than 50% of the grade if you do not use recursion for the main part of the program.

Deliverables

Turn in a single file, *matching.c*, over WebCourses that solves the specified problem. If you decide to make any enhancements to this program, clearly specify them in your header comment so the grader can test your program accordingly.