

# LAB 6 – STABLE MARRIAGE

CS 143

You may work individually or in groups of two

This program will give you practice with a complex data structure. You are going to write a program that solves a classic computer science problem known as the stable marriage problem. The input file is divided into men and women and the program tries to pair them up so as to generate as many marriages as possible that are all stable. A set of marriages is unstable if you can find a man and a woman who would rather be married to each other than to their spouses (in which case, the two would be inclined to divorce their spouses and marry each other).

The input file for the program will list all of the men, one per line, followed by an input line with just the word “END” on it, followed by all of the women, one per line, followed by another input line with just the word “END” on it. The men and women are numbered by their position in the input file. To make this easier, we have numbered starting at 0 (the first man is #0, the second man is #1, and so on; the first woman is #0, the second woman is #1, and so on). Each input line (except the two lines with “END”) has a name followed by a colon followed by a list of integers. The integers are the preferences for this particular person. For example, the following input line in the men's section:

```
Joe: 9 7 34 8 19 21 32 5 28 6 31 15 17 24
```

indicates that the person is named “Joe” and that his first choice for marriage is woman #9, his second choice is woman #7, and so on. Any women not listed are considered unacceptable to Joe. The data file has been purged of any impossible pairings where one person is interested in the other, but the other considers that person unacceptable. Thus, if a woman appears on Joe’s list, then Joe is acceptable to that woman.

There are many ways to approach the stable marriage problem. You are to implement a specific algorithm described below. This is the basic outline:

```
set each person to be free;
while (some man m with a nonempty preference list is free) {
    w = first woman on m's list;
    if (some man p is engaged to w) {
        set p to be free
    }
    set m and w to be engaged to each other
    for (each successor q of m on w's list) {
        delete w from q's preference list
        delete q from w's preference list
    }
}
```

Consider, for example, the following short input: (Note in the following examples Man 0 would be the first man's name in a real data file the first line which reads Man 0: 3 0 1 2 might be replaced by Joe: 3 0 1 2)

Man 0: 3 0 1 2	Woman 0: 3 0 2 1
Man 1: 1 2 0 3	Woman 1: 0 2 1 3
Man 2: 1 3 2 0	Woman 2: 0 1 2 3
Man 3: 2 0 3 1	Woman 3: 3 0 2 1

It doesn't matter where we start, so suppose we start with man 0. We engage him to woman 3 (the first choice on his list). Men 2 and 1 appear after man 0 in woman 3's list. Thus, we delete those connections.

Man 0: 3 0 1 2	Woman 0: 3 0 2 1	Man 0 and Woman 3 engaged
Man 1: 1 2 0	Woman 1: 0 2 1 3	
Man 2: 1 2 0	Woman 2: 0 1 2 3	
Man 3: 2 0 3 1	Woman 3: 3 0	

Notice that three things have changed: the list for man 1, the list for man 2 and the list for woman 3.

Suppose we next go to man 1. We engage him to woman 1. Man 3 appears after man 1 on woman 1's list, so we have to delete that connection, obtaining:

Man 0: 3 0 1 2	Woman 0: 3 0 2 1	Man 0 and Woman 3 engaged
Man 1: 1 2 0	Woman 1: 0 2 1	Man 1 and Woman 1 engaged
Man 2: 1 2 0	Woman 2: 0 1 2 3	
Man 3: 2 0 3	Woman 3: 3 0	

Notice that two things have changed: the list for man 3 and the list for woman 1.

Suppose we next go to man 2. We engage him to woman 1, which requires breaking woman 1's engagement to man 1 (making man 1 again available). Man 1 appears after man 2 on woman 1's list, so we break that connection, obtaining:

Man 0: 3 0 1 2	Woman 0: 3 0 2 1	Man 0 and Woman 3 engaged
Man 1: 2 0	Woman 1: 0 2	
Man 2: 1 2 0	Woman 2: 0 1 2 3	Man 2 and Woman 1 engaged
Man 3: 2 0 3	Woman 3: 3 0	

Suppose we next go to man 3. We engage him to woman 2. Nobody appears on woman 2's list after man 3, so we don't have to remove any connections:

Man 0: 3 0 1 2	Woman 0: 3 0 2 1	Man 0 and Woman 3 engaged
Man 1: 2 0	Woman 1: 0 2	
Man 2: 1 2 0	Woman 2: 0 1 2 3	Man 2 and Woman 1 engaged
Man 3: 2 0 3	Woman 3: 3 0	Man 3 and Woman 2 engaged

Now we go back to man 1 (because he's still free). We engage him to woman 2, which requires breaking woman 2's engagement to man 3 (making man 3 again available). Men 2 and 3 appear after man 1 on woman 2's list, so we break those connections, obtaining:

Man 0: 3 0 1 2	Woman 0: 3 0 2 1	Man 0 and Woman 3 engaged
Man 1: 2 0	Woman 1: 0 2	Man 1 and Woman 2 engaged
Man 2: 1 0	Woman 2: 0 1	Man 2 and Woman 1 engaged
Man 3: 0 3	Woman 3: 3 0	

Finally, the last free man is man 3. We engage him to woman 0. We could actually stop at this point, but according to the algorithm, we should notice that men 0, 2 and 1 all appear after man 3 on woman 0's list, so we break those connections:

Man 0: 3 1 2	Woman 0: 3	Man 0 and Woman 3 engaged
Man 1: 2	Woman 1: 0 2	Man 1 and Woman 2 engaged
Man 2: 1	Woman 2: 0 1	Man 2 and Woman 1 engaged
Man 3: 0 3	Woman 3: 3 0	Man 3 and Woman 0 engaged

This, then, is our stable marriage solution.

An interesting property of this simple stable marriage algorithm is that it favors one group over the other. The approach above favors the men at the expense of the women. In fact, the final result will give each man his best possible match for stable marriage situations, and will give each woman her worst possible match. But there is no reason that the algorithm can't be reversed with the women being favored over the men (just reversing the roles in the algorithm above).

You will also report in a column called “Choice” the relative position of the chosen partner in the original preference list and you will report the average of the choice column at the end of each sublist. For example, you will find the following line of output for the result that favors the men:

Name	Choice	Partner
-----		
William	6	Rachana

This indicates that William is paired with Rachana, and that Rachana was 6th on William's original list. The result that favors the women has this line of output:

Name	Choice	Partner
-----		
William	19	Caroline

This indicates that William is paired with Caroline, and that Caroline was 19th on William's original list.

The stable marriage algorithm always converges on an answer, but it is possible that some people will end up without being paired (this is inevitable in the large sample input file where there are 40 men and 35 women). Remember that the original algorithm terminates when there are no men left who are free and have a nonempty preference list. People are unpaired in the final result if they run out of choices on their preference lists. Be careful not to report a “choice” value for these people and don't include them in the calculation of the overall choice value.

**A sample input file has been posted on Canvas, along with a file that shows what would be displayed as output to the console. Your program MUST read a file of exactly the format as described above and of the sample, and your output should look similar to the output in the sample output file.**

Please write your program to request a filename from the user, so that I can compile your code once and test it with multiple input files.