

- 1) Study Stable Marriable Problem and its solution discussed in class. This assignment is relatively straightforward as you are required to implement Gale-Shapley Algorithm in Python 3 **using the approach and data structures described in the first two Chapters of Kleinberg and Tardos text and discussed in class/slides**. Make sure to document your code in such a way that the correspondence between the pseudocode steps and the program code is transparent. (You may also watch the following video explaining the need to generalize the algorithm further to deal with the practical problem of matching Hospitals and Residency Applicants we used as motivating example: <https://www.youtube.com/watch?v=kvfgGmemdA>. Note that this differs from Stable Marriage Problem in that a hospital can have several slots and not all residency applicants will get a position.)

In your solution, make sure:

- (a) The input for an n -men- n -women problem is provided as a sequence of $(2n+1)$ -lines (showing the value n on the first line, followed by the encodings of the preference lists – n lines for men and n lines for women) with each line containing *spaces-separated* list of $n+1$ totally ordered values formatted as follows:

```
n
MAN_1  WOMAN_i1  WOMAN_i2 ...
...
MAN_n  WOMAN_in1 WOMAN_in2 ...
WOMAN_1  MAN_j1  MAN_j2 ...
...
WOMAN_n  MAN_jn1 MAN_jn2 ...
```

For a sample example, consider:

```
3
Albert  Diane  Emily  Fergie
Bradley  Emily  Diane  Fergie
Charles  Diane  Emily  Fergie
Diane  Albert  Bradley  Charles
Emily  Albert  Bradley  Charles
Fergie  Albert  Bradley  Charles
```

- (b) The format of the output file is as follows showing the matching of MEN to WOMEN as:

```
MAN_1  WOMAN_k1
MAN_2  WOMAN_k2
...
MAN_n  WOMAN_kn
p
```

In addition, you should count the number of proposals offered, which is the same as the number of iterations, the number p on the last line. (In general, this number is not fixed due to the non-deterministic nature of the algorithm and so depends on the sequence of proposals offered. So do not be wedded to the number 3 below!)

For the above example, the output solution can be:

```
Albert  Diane
```

- (c) NAMING CONVENTIONS: You must name input file, output file, and the Python program file as follows: **Input.txt**, **Output.txt**, **assignment1.py**. Make sure that your Python program receives only one argument, the input file, from the command line and generates the output file in the current working directory, following the given naming convention. Note that the output file name can be automatically generated in accordance with this convention in the same directory. *Follow these naming conventions and file extensions.*

In addition to the above three files, you must also create a separate folder called **Tests** in the current working directory that contains adequate additional tests you have carried out to convince yourself that your code works correctly. Name these files **Input1.txt**, **Output1.txt**, **Input2.txt**, **Output2.txt**, **Input3.txt**, **Output3.txt**, ...

*In addition to the above tests, you must include a worst-case test for n = 4 in **Tests** with files named **Input0.txt** and **Output0.txt**.* This is mandatory.

In your program, include pseudo-code of the Gale Shapley algorithm as a comment to see the correspondence between the algorithm and the implementing code. As this is a team project, you must program in Python 3 as one of your team members should know Python. *Further, it is the team's collective responsibility to develop, test, and submit the code as any one of you can be asked to run it or explain it to us if needed.*

- (d) Additionally, include a function called *stabilityChecker* in a file called **stabilityChecker.py** that can read in two files **Input.txt** and your output file **OutputToBeVerified.txt**, check whether the assignment of men to women is stable, and writes out another file called **Verified.txt** containing the word **stable** or **unstable** as the only line.

In addition to the above four files (one Python code and three txt files), you must create a separate folder called **VerifierTests** in the current working directory that contains adequate additional tests you have carried out to be convinced that your stability checker works correctly. Name these files **Input1.txt**, **OutputToBeVerified1.txt**, **Verified1.txt**, **Input2.txt**, **OutputToBeVerified2.txt**, **Verified2.txt**, ...

- 2) Do Problem 4 in Chapter 2 on Pages 67-68 of the Kleinberg and Tardos text. Justify your ordering of the growth rate functions in each case. You are encouraged to try more exercises and discuss with your peers to improve your understanding.

TURNIN NOT REQUIRED AS THE ASSIGNMENT IS NOT GRADED. If you want specifics on the solution format that was required last time when the assignment was graded, here are the details:

Create one zip archive per team of upto 3 students containing (i) program file **assignment1.py** (with team member name(s) and email address(es) clearly given in the source code file(s) and any **README.txt**), **Input.txt**, **Output.txt**, as well as additional sample inputs and outputs (in the format discussed above, organized in a separate **Tests** directory/folder nested in the top-level directory), as well as (ii) program file **stabilityChecker.py**, **OutputToBeVerified.txt**, **Verified.txt**, as well as additional sample inputs and outputs (in the format discussed above, organized in a separate **VerifierTests** directory/folder nested in the top-level directory), to Assignment 1 DropBox on Pilot.

FYI: The **Output.txt** and **OutputToBeVerified.txt** in the top directory will be identical. However, other triplets of files in **VerifierTests** will be independent of what is in **Tests** because you will also be checking for unstable solutions that your Gale Shapley implementation will never generate.

Include only one submission per team called **StableMarriage.zip**. *Do not send redundant emails with your files to us beyond the Pilot submission as we will ignore such emails.*