REPORT

The algorithm functions as follows:

I. Initialization:

1. At first, every man and woman are free and unique.
2. Men have lists of women ranked according to their preferences.
3. Every woman has a ranking list of males based on her preferences.

II. Method of Proposal:

Although there is at least one single man who hasn't asked each woman on his list to marry him yet:

1. **Choose a Free Man:** Select any male who hasn't asked every lady to marry him and is available right now.
2. **Propose:** The man makes a proposal to the woman who is most deserving of his affections and to whom he has not yet made a proposal.
3. **Response:**
    - They get engaged if the woman is free. If the lady is already engaged, she evaluates both the new proposal and her current relationship.
    - She will call off her existing engagement and become engaged to the new man if she decides she prefers him.
    - The man who was previously engaged is released. She declines the new proposal if she thinks her present spouse is better.

III. Termination:

When no more free guys remain who haven't popped the question to every woman, the algorithm comes to a stop.

IV. Result:

In a stable matching produced by the algorithm, no two people would choose to be with their allocated mates.

Examples Explanation:

Let’s Consider 3 men_ (M1, M2, M3) and 3 women_ (W1, W2, W3) with following preference’s:

Men's_ Preference's:

Albert: Diane, Emily. Fergie

Bradley: Emily, Diane, Fergie

Charles: Diane, Emily, Fergie

Women's Preferences:

Diane: Albert, Bradley, Charles

Emily: Albert, Bradley, Charles

Fergie: Albert, Bradley, Charles

Step-by-Step Gale-Shapley Algorithm Execution:

Step 1: Initial State

All the men's (Albert, Bradley, Charles) & women's (Diane, Emily, Fergie) are unengaged.

Step 2: Proposals and Engagements

Diane is the woman Albert wants to propose to.

Diane agrees and is available.

Engagements: Emily, his first choice, gets proposed to by Diane Bradley to Albert.

Emily agrees and is available.

Albert ↔ Diane, Bradley → Emily are engaged.

Charles pops the question to Diane, his first choice.

Charles is rejected by Diane because she thinks Albert is better.

Albert ↔ Diane, Bradley → Emily are engaged.

Step 3: Continuing with Free Men

Charles makes Emily his next target for a proposal.

Emily rejects Charles because she thinks Bradley is better.

Engagements: Bradley → Emily, Albert ↔ Diane

Charles makes his final proposal to Fergie.

Fergie agrees and is available.

Engagements: Charles → Fergie, Bradley → Emily, and Albert → Diane

Final Matching:

Albert ↔ Diane

Bradley ↔ Emily

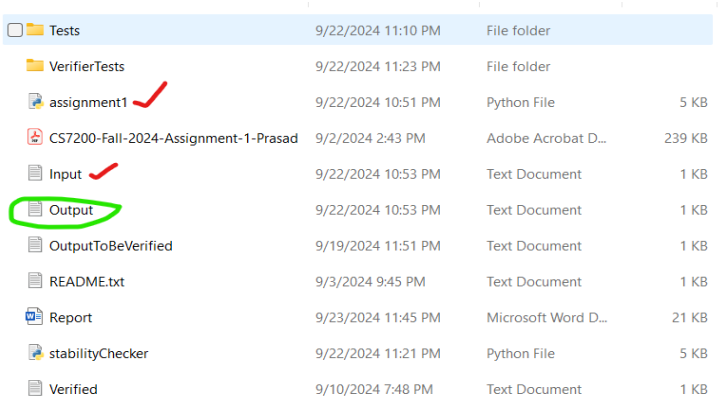
Charles ↔ Fergie

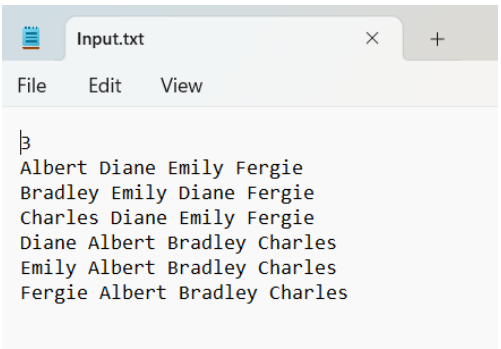
Examples

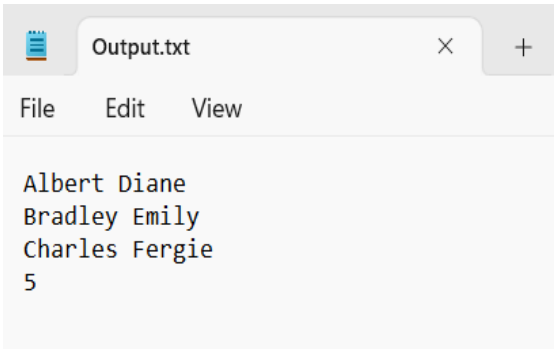
In my solution, taken **input.txt**, **output.txt** and **assignment1.py** python program files to check the output of the given input by only using single argument by using command line and helps in generating the output.txt file in the same working directory.

By using below Command, it automatically generates output.txt in same directory:

python assignment1.py Input.txt



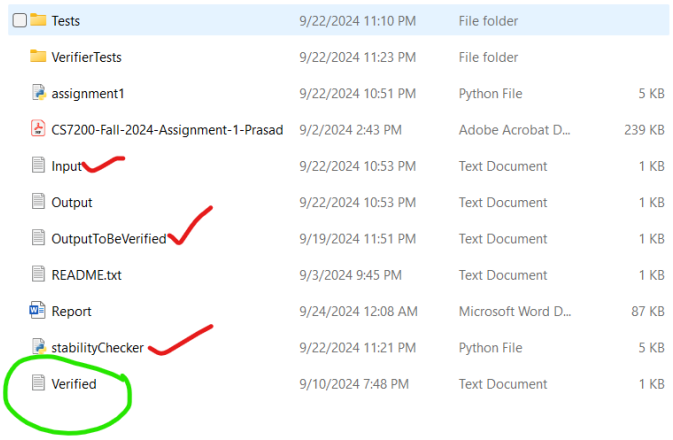


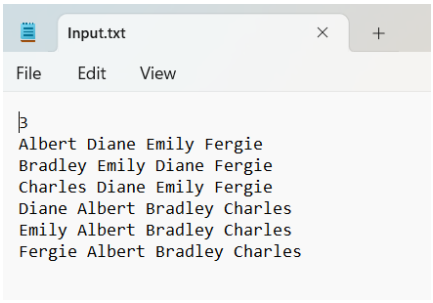


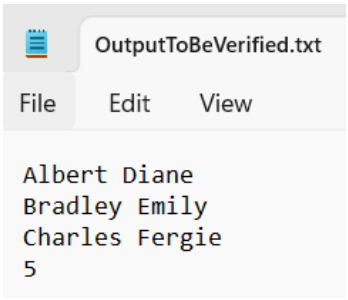
Now after output.txt file generated, need to generate the **verified.txt** file using **stabilityChecker.py** python program which checks the **output.txt** file generated and rename it to

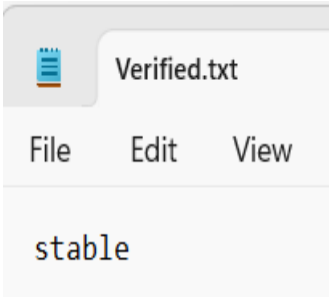
OutputToBeVerified.txt to check stability of output by giving below command line and gives single line word whether its stable or unstable.

python stabilityChecker.py Input.txt OutputToBeVerified.txt







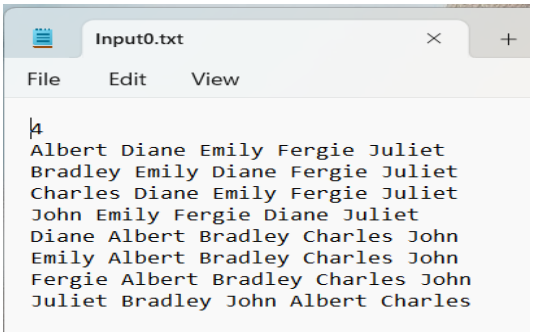


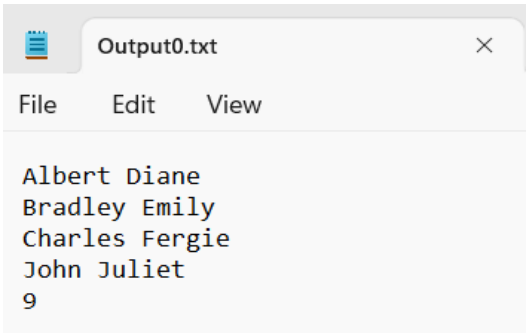
---

n = 4 (Example)→ Input0.txt, Output0.txt, OutputToBeVerified.txt, Verified0.txt

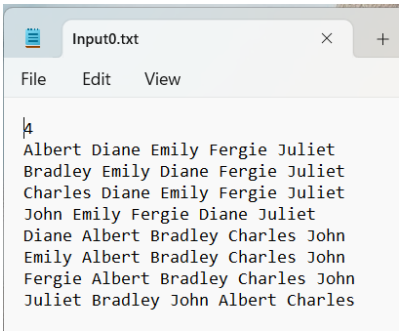
Command line:

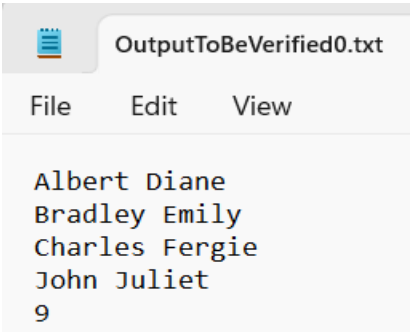
python assignment1.py Input0.txt

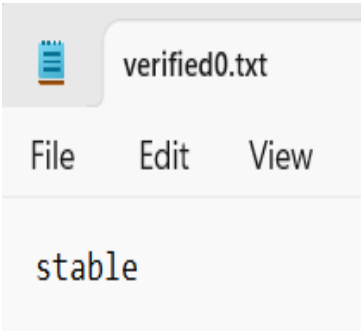




python stabilityChecker.py Input0.txt OutputToBeVerified0.txt

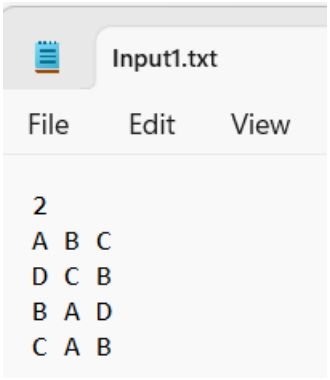


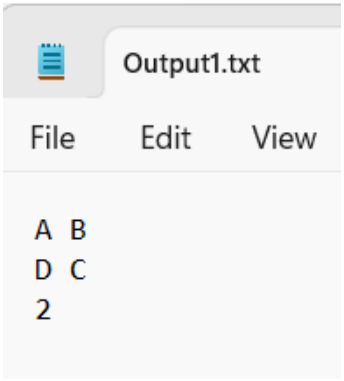




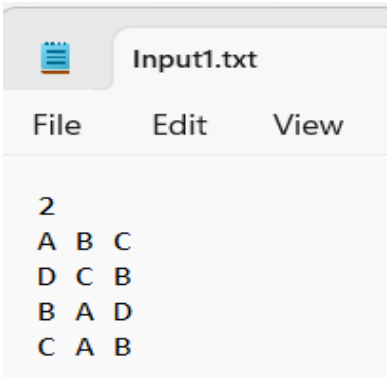
Examples:

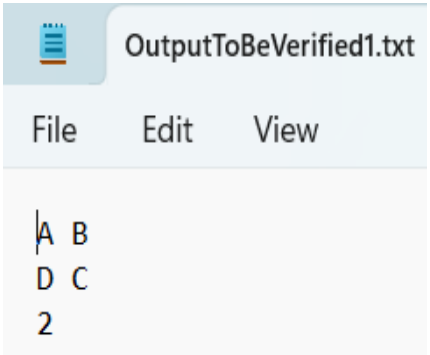
python assignment1.py Input1.txt

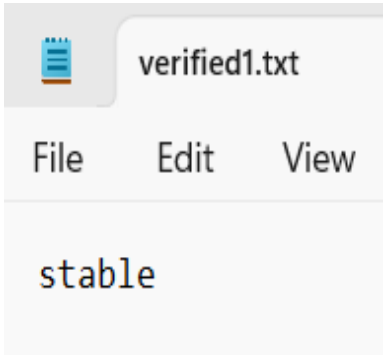




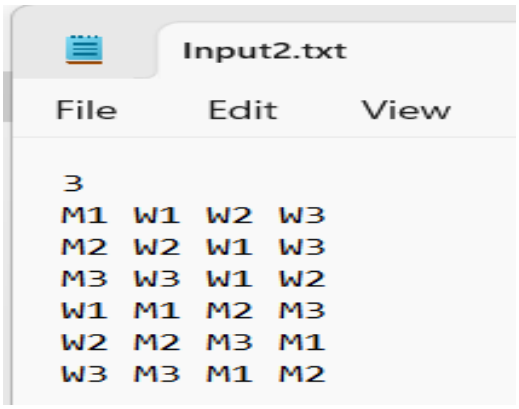
python stabilityChecker.py Input1.txt OutputToBeVerified1.txt

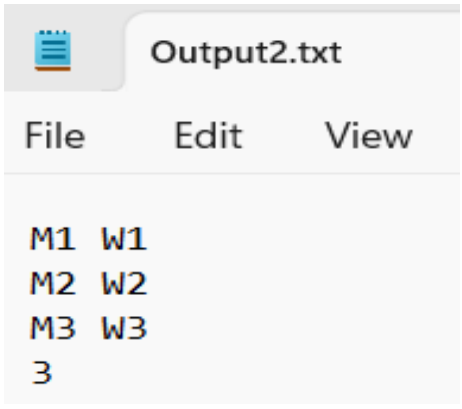




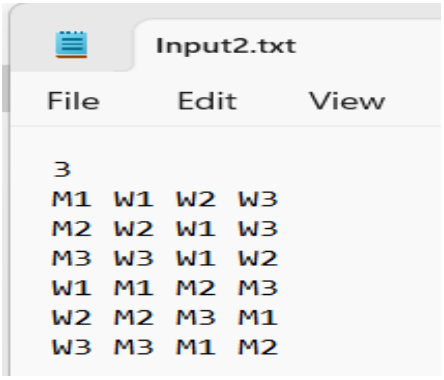


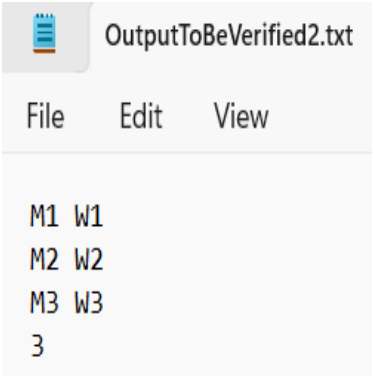
python assignment1.py Input2.txt

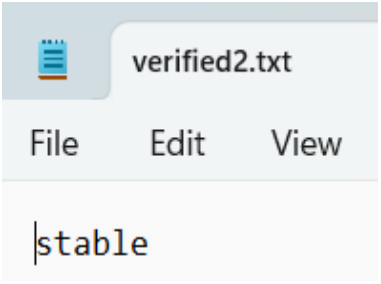




python stabilityChecker.py Input2.txt OutputToBeVerified2.txt

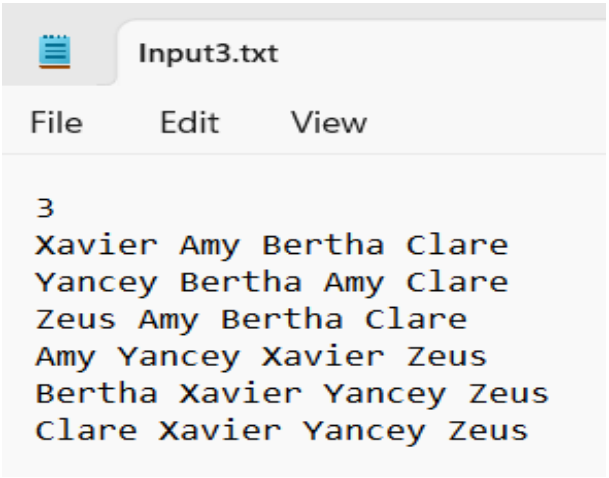


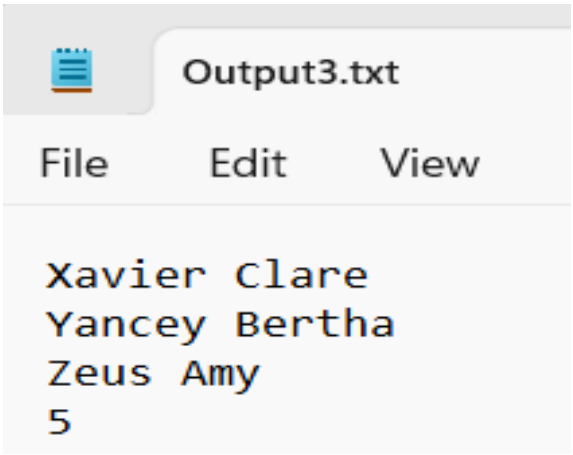




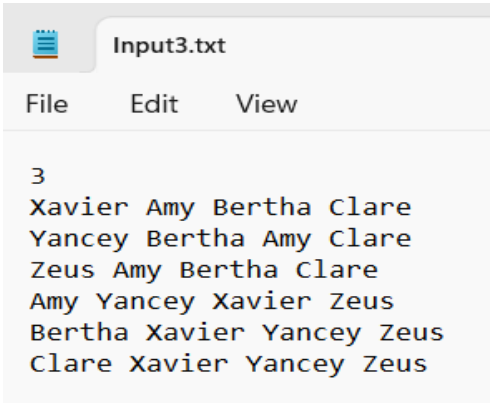
Unstable example(Testcase to check stabilityChecker.py)

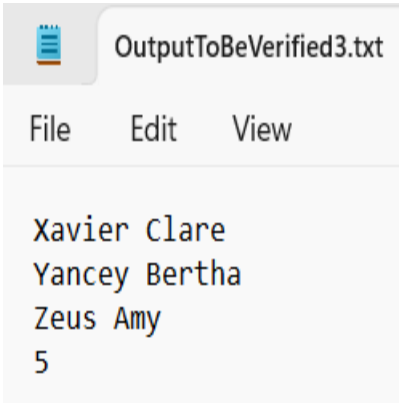
python assignment1.py Input3.txt

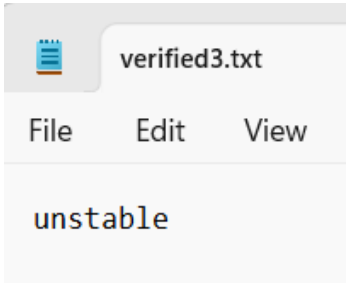




python stabilityChecker.py Input3.txt OutputToBeVerified3.txt

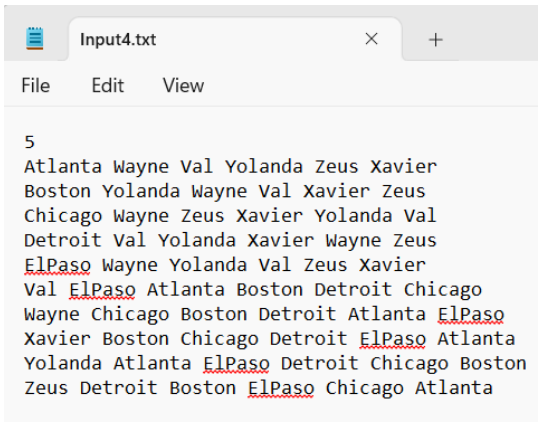


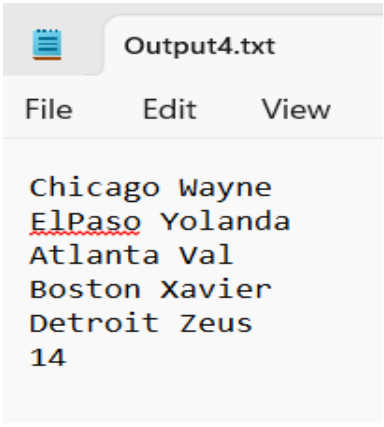




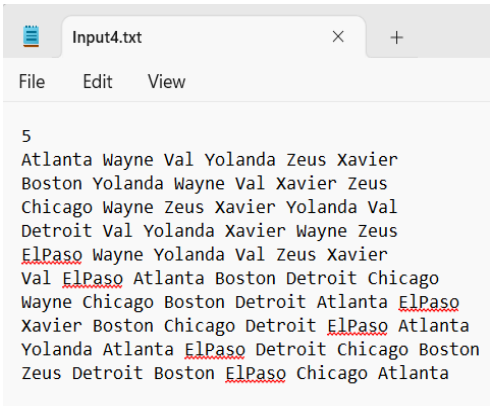
---

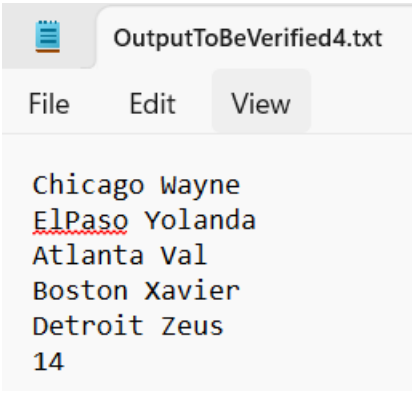
python assignment1.py Input4.txt

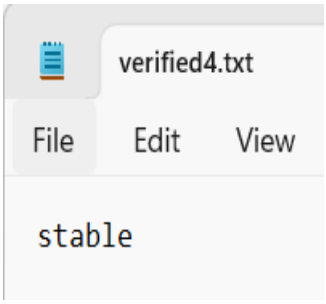




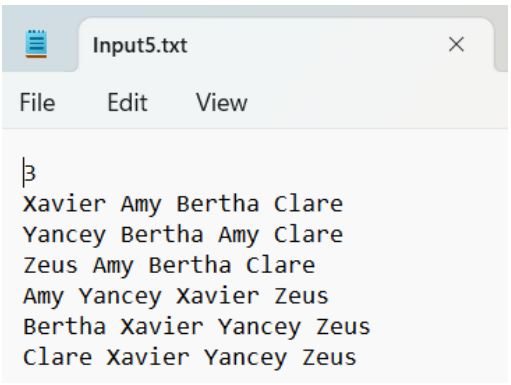
python stabilityChecker.py Input4.txt OutputToBeVerified4.txt

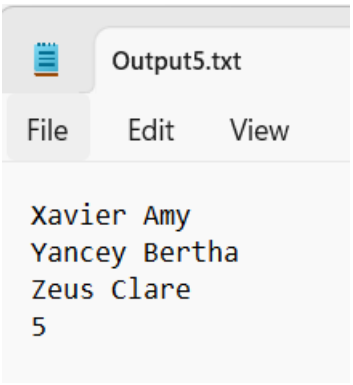




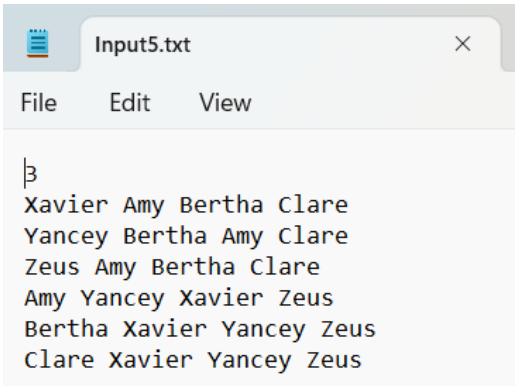


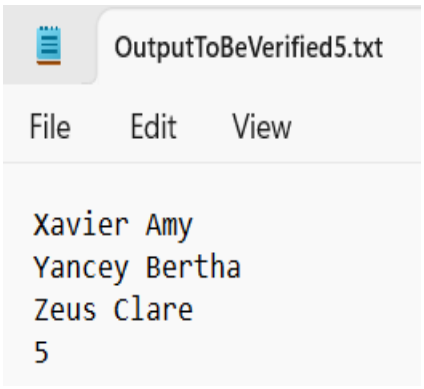
python assignment1.py Input5.txt

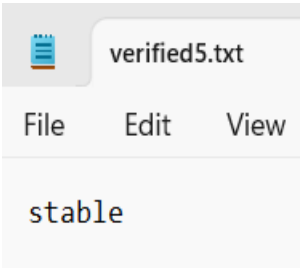




python stabilityChecker.py Input5.txt OutputToBeVerified5.txt

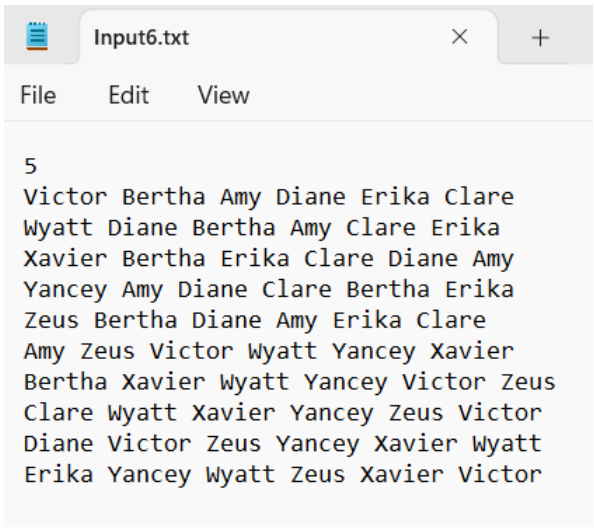


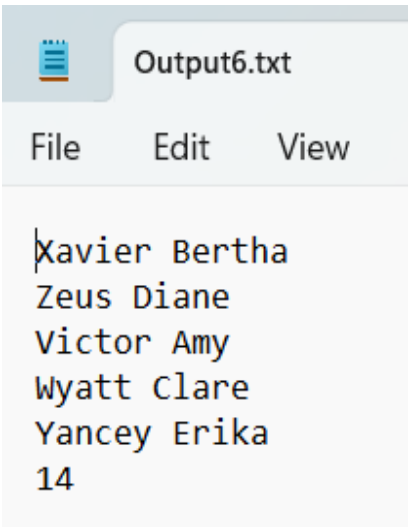




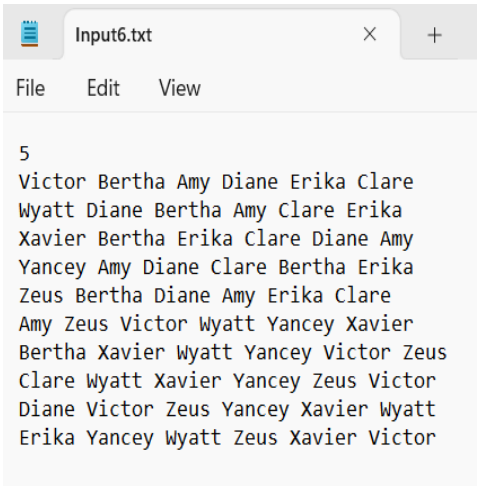
---

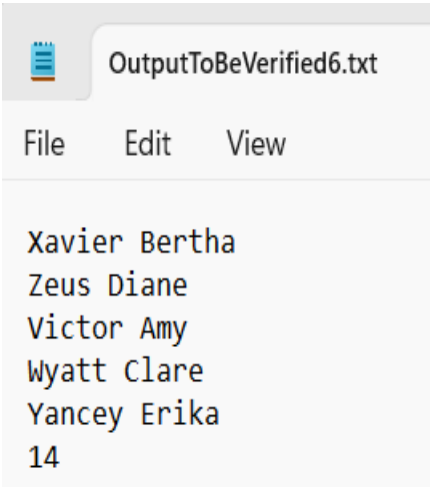
python assignment1.py Input6.txt

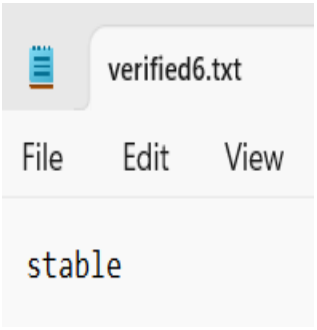




python stabilityChecker.py Input6.txt OutputToBeVerified6.txt

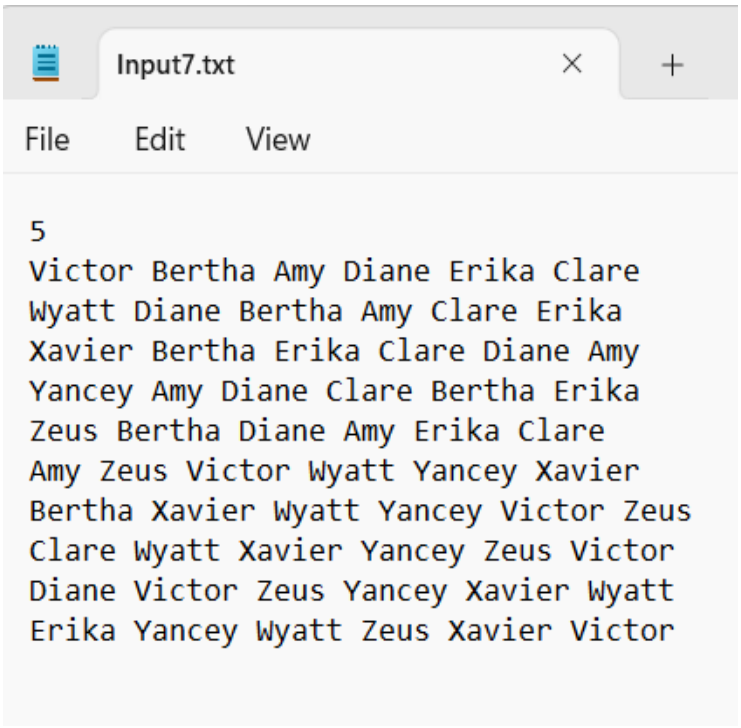


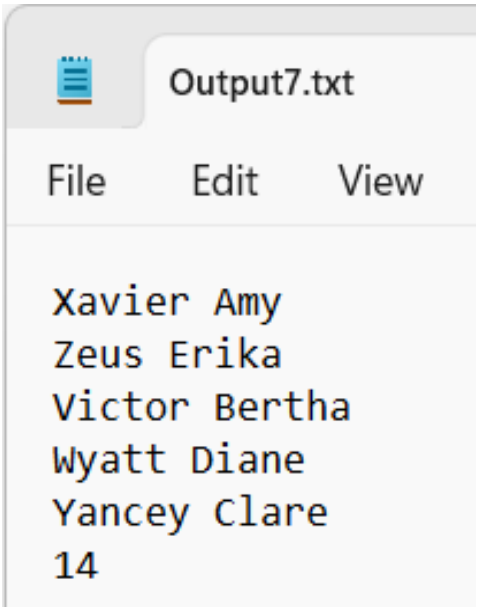




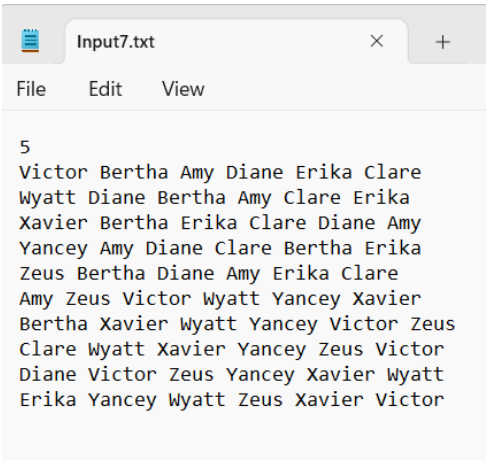
Unstable example(Testcase to check stabilityChecker.py)

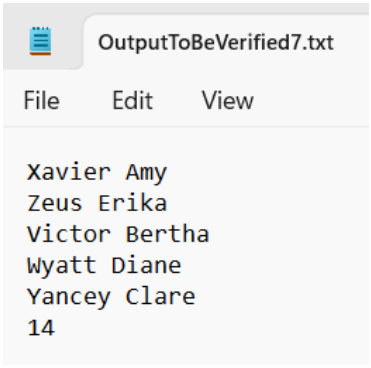
python assignment1.py Input7.txt

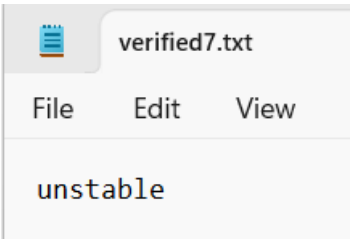




python stabilityChecker.py Input7.txt OutputToBeVerified7.txt







Results Screenshot's (assignment1.py and stabilitChecker.py)

```
C:\Users\madhu\Desktop\StableMarriage>python assignment1.py Input.txt
Results saved to Input.txt

C:\Users\madhu\Desktop\StableMarriage>python assignment1.py Input.txt
Results saved to Output.txt

C:\Users\madhu\Desktop\StableMarriage>python assignment1.py Input0.txt
Results saved to Output0.txt

C:\Users\madhu\Desktop\StableMarriage>python assignment1.py Input1.txt
Results saved to Output1.txt

C:\Users\madhu\Desktop\StableMarriage>python assignment1.py Input2.txt
Results saved to Output2.txt

C:\Users\madhu\Desktop\StableMarriage>python assignment1.py Input3.txt
Results saved to Output3.txt

C:\Users\madhu\Desktop\StableMarriage>python assignment1.py Input4.txt
Results saved to Output4.txt

C:\Users\madhu\Desktop\StableMarriage>python assignment1.py Input5.txt
Results saved to Output5.txt

C:\Users\madhu\Desktop\StableMarriage>python assignment1.py Input6.txt
Results saved to Output6.txt

C:\Users\madhu\Desktop\StableMarriage>
```

```
Microsoft Windows [Version 10.0.22631.3447]
(c) Microsoft Corporation. All rights reserved.

C:\Users\madhu\Desktop\StableMarriage\VerifierTests>python stabilityChecker.py Input.txt OutputToBeVerified.txt
Skipping invalid line: 5
Loaded Engagements Dictionary: {'Albert': 'Diane', 'Bradley': 'Emily', 'Charles': 'Fergie'}
Engagements Dictionary: {'Albert': 'Diane', 'Bradley': 'Emily', 'Charles': 'Fergie'}
Reversed Engagements Dictionary: {'Diane': 'Albert', 'Emily': 'Bradley', 'Fergie': 'Charles'}
Verifying engagement: Albert -> Diane
Verifying engagement: Bradley -> Emily
Verifying engagement: Charles -> Fergie
Verification result saved to verified.txt

C:\Users\madhu\Desktop\StableMarriage\VerifierTests>python stabilityChecker.py Input0.txt OutputToBeVerified0.txt
Skipping invalid line: 9
Loaded Engagements Dictionary: {'Albert': 'Diane', 'Bradley': 'Emily', 'Charles': 'Fergie', 'John': 'Juliet'}
Engagements Dictionary: {'Albert': 'Diane', 'Bradley': 'Emily', 'Charles': 'Fergie', 'John': 'Juliet'}
Reversed Engagements Dictionary: {'Diane': 'Albert', 'Emily': 'Bradley', 'Fergie': 'Charles', 'Juliet': 'John'}
Verifying engagement: Albert -> Diane
Verifying engagement: Bradley -> Emily
Verifying engagement: Charles -> Fergie
Verifying engagement: John -> Juliet
Verification result saved to verified0.txt

C:\Users\madhu\Desktop\StableMarriage\VerifierTests>python stabilityChecker.py Input1.txt OutputToBeVerified1.txt
Skipping invalid line: 2
Loaded Engagements Dictionary: {'A': 'B', 'D': 'C'}
Engagements Dictionary: {'A': 'B', 'D': 'C'}
Reversed Engagements Dictionary: {'B': 'A', 'C': 'D'}
Verifying engagement: A -> B
Verifying engagement: D -> C
Verification result saved to verified1.txt

C:\Users\madhu\Desktop\StableMarriage\VerifierTests>python stabilityChecker.py Input2.txt OutputToBeVerified2.txt
Skipping invalid line: 3
Loaded Engagements Dictionary: {'M1': 'W1', 'M2': 'W2', 'M3': 'W3'}
Engagements Dictionary: {'M1': 'W1', 'M2': 'W2', 'M3': 'W3'}
Reversed Engagements Dictionary: {'W1': 'M1', 'W2': 'M2', 'W3': 'M3'}
Verifying engagement: M1 -> W1
Verifying engagement: M2 -> W2
Verifying engagement: M3 -> W3
Verification result saved to verified2.txt
```

```
C:\Users\madhu\Desktop\StableMarriage\VerifierTests>python stabilityChecker.py Input3.txt OutputToBeVerified3.txt
Skipping invalid line: 5
Loaded Engagements Dictionary: {'Xavier': 'Clare', 'Yancey': 'Bertha', 'Zeus': 'Amy'}
Engagements Dictionary: {'Xavier': 'Clare', 'Yancey': 'Bertha', 'Zeus': 'Amy'}
Reversed Engagements Dictionary: {'Clare': 'Xavier', 'Bertha': 'Yancey', 'Amy': 'Zeus'}
Verifying engagement: Xavier -> Clare
Unstable pair found: Xavier prefers Amy, and Amy prefers Xavier over Zeus.
Verification result saved to verified3.txt

C:\Users\madhu\Desktop\StableMarriage\VerifierTests>python stabilityChecker.py Input4.txt OutputToBeVerified4.txt
Skipping invalid line: 14
Loaded Engagements Dictionary: {'Chicago': 'Wayne', 'ElPaso': 'Yolanda', 'Atlanta': 'Val', 'Boston': 'Xavier', 'Detroit': 'Zeus'}
Engagements Dictionary: {'Chicago': 'Wayne', 'ElPaso': 'Yolanda', 'Atlanta': 'Val', 'Boston': 'Xavier', 'Detroit': 'Zeus'}
Reversed Engagements Dictionary: {'Wayne': 'Chicago', 'Yolanda': 'ElPaso', 'Val': 'Atlanta', 'Xavier': 'Boston', 'Zeus': 'Detroit'}
Verifying engagement: Chicago -> Wayne
Verifying engagement: ElPaso -> Yolanda
Verifying engagement: Atlanta -> Val
Verifying engagement: Boston -> Xavier
Verifying engagement: Detroit -> Zeus
Verification result saved to verified4.txt
```

```
C:\Users\madhu\Desktop\StableMarriage\VerifierTests>python stabilityChecker.py Input5.txt OutputToBeVerified5.txt
Skipping invalid line: 5
Loaded Engagements Dictionary: {'Xavier': 'Amy', 'Yancey': 'Bertha', 'Zeus': 'Clare'}
Engagements Dictionary: {'Xavier': 'Amy', 'Yancey': 'Bertha', 'Zeus': 'Clare'}
Reversed Engagements Dictionary: {'Amy': 'Xavier', 'Bertha': 'Yancey', 'Clare': 'Zeus'}
Verifying engagement: Xavier -> Amy
Verifying engagement: Yancey -> Bertha
Verifying engagement: Zeus -> Clare
Verification result saved to verified5.txt

C:\Users\madhu\Desktop\StableMarriage\VerifierTests>python stabilityChecker.py Input6.txt OutputToBeVerified6.txt
Skipping invalid line: 14
Loaded Engagements Dictionary: {'Xavier': 'Bertha', 'Zeus': 'Diane', 'Victor': 'Amy', 'Wyatt': 'Clare', 'Yancey': 'Erika'}
Engagements Dictionary: {'Xavier': 'Bertha', 'Zeus': 'Diane', 'Victor': 'Amy', 'Wyatt': 'Clare', 'Yancey': 'Erika'}
Reversed Engagements Dictionary: {'Bertha': 'Xavier', 'Diane': 'Zeus', 'Amy': 'Victor', 'Clare': 'Wyatt', 'Erika': 'Yancey'}
Verifying engagement: Xavier -> Bertha
Verifying engagement: Zeus -> Diane
Verifying engagement: Victor -> Amy
Verifying engagement: Wyatt -> Clare
Verifying engagement: Yancey -> Erika
Verification result saved to verified6.txt

C:\Users\madhu\Desktop\StableMarriage\VerifierTests>python stabilityChecker.py Input7.txt OutputToBeVerified7.txt
Skipping invalid line: 14
Loaded Engagements Dictionary: {'Xavier': 'Amy', 'Zeus': 'Erika', 'Victor': 'Bertha', 'Wyatt': 'Diane', 'Yancey': 'Clare'}
Engagements Dictionary: {'Xavier': 'Amy', 'Zeus': 'Erika', 'Victor': 'Bertha', 'Wyatt': 'Diane', 'Yancey': 'Clare'}
Reversed Engagements Dictionary: {'Amy': 'Xavier', 'Erika': 'Zeus', 'Bertha': 'Victor', 'Diane': 'Wyatt', 'Clare': 'Yancey'}
Verifying engagement: Xavier -> Amy
Unstable pair found: Xavier prefers Bertha, and Bertha prefers Xavier over Victor.
Verification result saved to verified7.txt

C:\Users\madhu\Desktop\StableMarriage\VerifierTests>
```