Assignment: asg1

Name: satish Dodda

ULID: 809961786

Secret Directory: behappy

Methods discerptions:

**computeGaleShapley(men, woman, m,w)->dict:**

This method will accept a men preference list, woman preference list and two variables m is for men and w is for women and the method will return a stored dictionary of stable matches generated by the Gale-Shapley algorithm

**printLIst(men, women ,count, m, w):**

This method will accept a men preference list women preference list count is used to print the set numbers and m and w flags for men and woman

This method will call the computeGaleShapley method after calling it will convert the data from a dictionary to a list of tuples and then sorting the data for correcting ordering of the data and displaying the data

**Algorithm Working**

First there will be a list of unmarried men and women, and they have their preference list’s

I have stored all unmarried man/women in not\_at\_paired dictionary

And I have taken two other dictionary’s men\_status, woman\_status to keep track of the men and women engagement status and with whom they are engaged

If the length of not\_at\_paired is greater than 0 then it means still some men/woman are not engaged so while loop will start run

And a for is starts iterate over men/woman preference list

If men is not engaged and the preferred woman is also not engaged then remove the men/woman from the not\_at\_pair dictionary and update the men\_status and woman\_status values list first value as 1 which means engaged and the second value as the index of the preferred men/women

If men is not engaged and the woman is already engaged then checking the current man index in the woman preference list and the previous men’s index in the woman preference list if the current man index is less than then previous men index then break the engagement with previous men and engage with the current man

So if we break the engagement then previous men will again added to the not\_at\_pair dictionary and updating men\_status with 0 and deleting the current engaged men from the not\_at\_pair list

This will continue until no men/woman left in the not\_at\_pair list and finally return the dictionary which contains perfect stable matches

**Data Structures used :**

**not\_at\_paired : i**t is the dictionary to store men/woman who is not at engaged

**men\_status:**  it is the dictionary to store the men engaged status and current engaged woman index in a list

**woman\_status** : it is the dictionary to store the woman engaged status and current engaged men index in a list

**men and woman:**  it is the dictionary to store men and woman preference list

**Time Complexity:**

Over all time complexity of the algorithm is O(N^2) where n is the number of men/ woman in the worst case every men is checked with every other woman

**Space Complexity:**

Over all space complexity of the algorithm is O(N) where n is the number of men/woman to store men\_status /woman\_status and not\_at\_pair dictionarys

**Difficulties:**

**The only difficulty I have faced is when a woman is breaking a relationship with men then men have to start processing from the next preferred woman in his preferred list at that point men are at random points in there preference list’s for next woman so inordered to keep track of there current preferred woman index i come up with men\_status dictionary to store current preferred woman index**