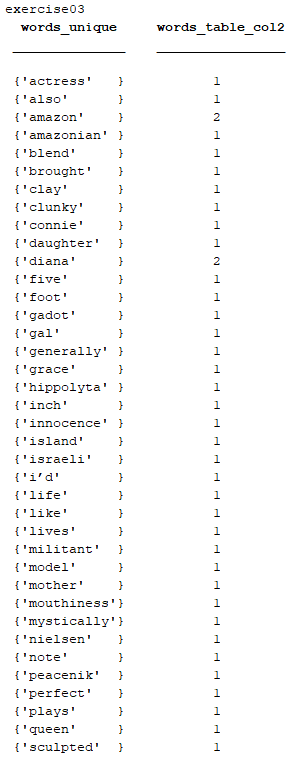
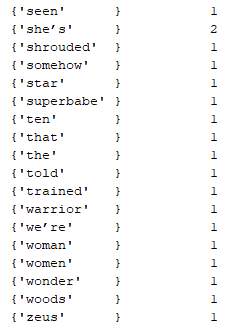
**CSE 408 - Homework 1 Results**

|  | **Exercise One** | | **Exercise Two** |
| --- | --- | --- | --- |
| Term | Similarity to ‘goodfellow’ by common letters | Similarity to ‘goodfellow’ by common letters with correct letter position considered | Similarity to ‘goodfellow’ by using stringdist |
| hello | 3 | 0 | 6 |
| goodbye | 4 | 4 | 6 |
| hola | 2 | 1 | 8 |
| Hello hellen | 3 | 0 | 8 |
| helmet | 2 | 0 | 8 |
| hellorheaven | 3 | 0 | 10 |
| hillsboro | 2 | 1 | 9 |
| Say hello | 3 | 4 | 6 |
| myfellow | 5 | 0 | 4 |

**1.5)** I am not exactly sure what this asks me to explain, but I implemented this design by simply looping through the indices of each character in the list of strings and comparing it to the character with the same position in ‘goodfellow’, until one of the strings was complete. This is a much more accurate way to tell similarity than simply finding similar letters. The bonus to this method is twofold; first, the algorithm takes into account strings of multiple characters that are similar (‘good’ and ‘goodfellow’ are more similar than ‘doog’ and ‘goodfellow’), second this method takes into account the number of times a letter is similar (‘good’ and ‘goodfellow’ produce ‘good’ instead of ‘god’).

**3)**

****

****