CS 432 Assignment #6
Larry Montgomery 01093132

Question 1:

For the first section I adjusted the code of finding the similar movies using the pearson correlation to be used with users. I first converted the user data to a discrete form, a ‘1’ if matched and a ‘0’ if it didn’t match my profile. After using this with the pearson correlation I was given my 3 matches “864”, “645” and “625”. I honestly haven’t seen a few of the movies so I had to look them up for reference. The lisst are below.

```
Least Favorite Movies:

user: ('864', 3)
movie 0:  In the Mouth of Madness (1995)
movie 1:  Land Before Time III: The Time of the Great Giving (1995) (V)
movie 2:  Road to Wellville, The (1994)


user: ('645', 3)
movie 0:  Jungle2Jungle (1997)
movie 1:  Saint, The (1997)
movie 2:  In & Out (1997)


user: ('625', 3)
movie 0:  Birdcage, The (1996)
movie 1:  Broken Arrow (1996)
movie 2:  Con Air (1997)
```

```
Favorite Movies:

user: ('864', 3)
movie 0:  Wrong Trousers, The (1993)
movie 1:  Wizard of Oz, The (1939)
movie 2:  Willy Wonka and the Chocolate Factory (1971)


user: ('645', 3)
movie 0:  Young Frankenstein (1974)
movie 1:  Wings of Desire (1987)
movie 2:  Touch of Evil (1958)


user: ('625', 3)
movie 0:  Wrong Trousers, The (1993)
movie 1:  Terminator 2: Judgment Day (1991)
movie 2:  Star Wars (1977)
```

From these I was put it quite a predicament, I really like the favorites of 625, inparticular Star Wars and Terminator, but he really disliked a good movie in my opinion Con Air. I know it is not common for people to enjoy it but I do. I felt that I would be better represented by user ‘645’.

Question 2:

To find the least and most correlated user I adjusted the code from the recommendations.py from the collective intelligence book. The code would now return the worst matches, basically I removed the reversing of the sorted data, since sort in python sorts smallest (least rated) to largest (best rating). The method is below as well as the correlation of the other users to my substitute.

```
Top Recommendation
 [(5.000000000000001, 'Delta of Venus (1994)'), (5.0, 'They Made Me a Criminal (1939)'), (5.0, 'Star Kid (1997)'), (5.0, "Someone Else's America (1995)"), (5.0, 'Santa wit
Muscles (1996)')]

 Worst Recommendation
 [(1.0, '3 Ninjas: High Noon At Mega Mountain (1998)'), (1.0, 'American Strays (1996)'), (1.0, "Amityville 1992: It's About Time (1992)"), (1.0, 'Amityville: A New Generat
n (1993)'), (1.0, 'Amityville: Dollhouse (1996)')]
```

```
def worstMatches(
    prefs,
    person,
    n=5,
    similarity=sim_pearson,
):
    '''
    Returns the worst matches for person from the prefs dictionary.
    Number of results and similarity function are optional params.
    '''

    scores = [(similarity(prefs, person, other), other) for other in prefs
              if other != person]
    scores.sort()
    #scores.reverse()
    return scores[0:n]
```

Question 3:

To computer the rating of all the films and produce the top 5 worst and top 5 best. I used the recommendations.py and modified the sorting of the results. I used the getRecommendations method as seen below to obtain the dictionary of correlation values and I reversed the dictionary results when I wanted the highest values, based on the same reason I previded in the last section. This produced the two lists of recommendations. You can see both the code and lists below.

```
def get_worst_recomendations(id, data):
    new_data = recommendations.getRecommendations(data, id)
    new_data.reverse()
    print('\n Worst Recommendation\n',new_data[0:5])

def get_top_recomendations(id,data):
    output = recommendations.getRecommendations(data, id)
    print('\n Top Recommendation\n',output[0:5])
```

```
 Top Recommendation
 [(5.000000000000001, 'Delta of Venus (1994)'), (5.0, 'They Made Me a Criminal (1939)'), (5.0, 'Star Kid (1997)'), (5.0, "Someone Else's America (1995)"), (5.0, 'Santa wit
Muscles (1996)')]

 Worst Recommendation
 [(1.0, '3 Ninjas: High Noon At Mega Mountain (1998)'), (1.0, 'American Strays (1996)'), (1.0, "Amityville 1992: It's About Time (1992)"), (1.0, 'Amityville: A New Generat
n (1993)'), (1.0, 'Amityville: Dollhouse (1996)')]
```

Question4:

When I selected my favorite film and least favorite film I wasn't sure what the outcome could be. I felt that the results were in the same category but not exactly what I thought were the best matches. For example I would think that Schindler's List would be like more popular movies like Casablank, but I started to reflect I think that was personal bias. I looked up these films, most not ever seeing before and after some analysis many of them not all matched the style of film I had looked for. This shows me that with a large enough data set we can use collective intelligence to find similarities without having personal knowledge of the specific topic. So yes I would say it did a good job at classifying.

```
Films matching Spice World (1997) are:  [(0, '1-900 (1994)'), (0, '8 1/2 (1963)'), (0, '8 Seconds (1994)'), (0, 'Across the Sea of Time (1995)'), (0, 'Adventures of Pinocchio, The (1996)')]


Films Not matching Spice World (1997) are:  [(1.0, 'unknown'), (1.0, 'Wonderland (1997)'), (1.0, "Wes Craven's New Nightmare (1994)"), (1.0, 'Vermin (1998)'), (1.0, 'Under Siege 2: Dark Territory (1995)')]
```

```
Films matching Schindler's List (1993) are:  [(1.0, 'Wings of Courage (1995)'), (1.0, 'Spanish Prisoner, The (1997)'), (1.0, "Some Mother's Son (1996)"), (1.0, 'Shopping (1994)'), (1.0, 'Search for One-eye Jimmy, The (1996)')]


Films matching Schindler's List (1993) are:  [(0, 'American Strays (1996)'), (0, 'August (1996)'), (0, 'B. Monkey (1998)'), (0, 'Big Bang Theory, The (1994)'), (0, 'Bird of Prey (1996)')]
```

```python
def find_corerelated_films(data):
    favorite_film = "Schindler's List (1993)"
    least_film = 'Spice World (1997)'
    print('here')
    data_dict = recommendations.calculateSimilarItems(data)
    second_dict = recommendations.calculateNotSimilarItems(data)
    print(f'\n Films matching {favorite_film} are: ', data_dict[favorite_film], '\n')
    print(f'\n Films matching {favorite_film} are: ', second_dict[favorite_film], '\n')

    print(f'\n Films matching {least_film} are: ', second_dict[least_film], '\n')
    print(f'\n Films Not matching {least_film} are: ', data_dict[least_film], '\n')
```