Mini-Project: User-Based Recommendation System

Let's say we have 6 users, and they have rated 8 different movies on a scale of 1 to 10. Note that not all users have rated all movies.

```
UserMovieRatings = {
  'Amy': {'Family Plot':10, 'Rebecca':5, 'Spellbound':9, 'Star Trek':6},
  'Bill': {'Apocalypto':8, 'Braveheart':3, 'Rebecca':10, 'Spellbound':5, 'Star Trek':7},
  'Cathy': {'Spaceballs':7, 'The Ice Storm':4, 'Family Plot':5, 'Rebecca':9, 'Spellbound':1},
  'Dave': {'Braveheart':5, 'Rebecca':7, 'Spellbound':4},
  'Ernie': {'Apocalypto':3, 'Braveheart':8, 'Rebecca':1, 'Star Trek':7},
  'Fiona': {'The Ice Storm':3, 'Family Plot':10, 'Rebecca':6, 'Spellbound':10}}
```

You can build a simple User-Based Recommendation System as follows:

- Let's say you want to make recommendations for UserX
- Given a UserX, you can find the most similar user or the "nearest neighbor" of UserX by calculating the manhattan distance between UserX and every other user (not including UserX).
- The person with the smallest manhattan distance is considered the most similar user. Let's call this person UserXNN.
- You can now find recommendations for UserX by considering all the movies that UserXNN has rated but that UserX has not.

You have been provided with a framework for this MiniProject: "MiniProject – Framework.py".

- The Framework defines a function called **manhattanD** which takes two rating dictionaries **ratings1D** and **ratings2D** as parameters and returns the Manhattan Distance between the two dictionaries.
- Pseudo-code has been provided to you in the framework. So, you essentially need to plug in appropriate code for steps 1 through 5 in the framework.

Hints:

- You can use the last exercise in ExerciseSet7 as a starting point for the mini project.
- The next page provides intermediate and final answers for each user. You can use this to check your work.

Some things to keep in mind as you code:

- Make your code readable – for instance, use meaningful variable names and comments.
- Make your code elegant – for instance, balance the number of variables you introduce – too many or too few make your code difficult to debug, read, and maintain.
- Make your output readable and user-friendly

Original data (that has been provided to you as a nested dictionary above):
6 users have rated 8 different movies on a scale of 1 to 10.

Ratings

	Amy	Bill	Cathy	Dave	Ernie	Fiona
Apocalypto		8			3	
Braveheart		3		5	8	
Family Plot	10		5			10
Rebecca	5	10	9	7	1	6
Spaceballs			7			
Spellbound	9	5	1	4		10
Star Trek	6	7			7	
The Ice Storm			4			3

User-User similarities based on manhattan distances (that you will need to write code for):
The highlighted cells indicate the "nearest neighbor" (user with the smallest manhattan distance) for each user. For instance, Bill's nearest neighbor is Cathy (since she has the smallest manhattan distance from Bill along that row).

Similarity Matrix

	Amy	Bill	Cathy	Dave	Ernie	Fiona
Amy	-	10	17	7	5	2
Bill	10	-	5	6	19	9
Cathy	17	5	-	5	8	18
Dave	7	6	5	-	9	7
Ernie	5	19	8	9	-	5
Fiona	2	9	18	7	5	-

Final recommendations for each user (that you will need to write code for):
Recommendations are based on those movies that each user's nearest neighbor has rated, and that the user has not rated. For instance, Bill's nearest neighbor is Cathy. Recommendations for Bill therefore include those movies that Cathy has rated but that Bill has not.

Recommendations

Amy	[('The Ice Storm', 3)
Bill	[('Spaceballs', 7), ('Family Plot', 5), ('The Ice Storm', 4)
Cathy	[('Braveheart', 5)
Dave	[('Spaceballs', 7), ('Family Plot', 5), ('The Ice Storm', 4)
Ernie	[('Family Plot', 10), ('Spellbound', 9)
Fiona	[('Star Trek', 6)