Q1 Neighbors

1 Point

Given the graph in Problem 1a, after it is completed, what will the following expression return:

```
practice_graph.neighbors("A")
```

- (●) a data structure containing "B" and "C"
- ( ) a data structure containing ("A", "B") and ("A", "C")
- ( ) a data structure containing ("A", "B"), ("A", "C"), ("B", "A"), and ("C", "A")
- ( ) a data structure containing "B", "C", "D", "E", and "F"

EXPLANATION

Technically it returns a dict_keyiterator, that we often wrap in a call to list() to convert it into a list or in a call to set() to convert it into a set.

Q2 Best Friend

1 Point

Given the completed graph shown in Problem 1a, what is the **best** friend recommendation for node F based on the "recommend by number of common friends" algorithm.

- ( ) B and E (tied)
- ( ) C
- ( ) A, B, and E (tied)
- ( ) C & D (tied)

- [x] B
- [ ] A

EXPLANATION

B has two friends in common with F (C & D). A and E have one friend each in common with F. Thus B is the better recommendation based on this algorithm. C and D also have one friend in common with F, but they are already friends with F so the algorithm will not return them as possibilities.

Q3 Friends of Friends

1 Point

Given the complete Romeo and Juliet graph, what does this expression evaluate to:

```
friends_of_friends(rj, "Juliet")
```

- [ ] `['Friar Laurence', 'Romeo', 'Tybalt', 'Capulet']` with the elements in any order
- [ ] `{'Friar Laurence', 'Romeo', 'Tybalt', 'Capulet'}` with the elements in any order
- [x] `{'Mercutio', 'Paris', 'Escalus', 'Benvolio', 'Montague'}` with the elements in any order
- [ ] `['Mercutio', 'Paris', 'Escalus', 'Benvolio', 'Montague']` with the elements in any order
- [ ] `{'Mercutio', 'Paris', 'Escalus', 'Friar Laurence', 'Romeo', 'Benvolio', 'Montague', 'T` with the elements in any order
- [ ] `['Mercutio', 'Paris', 'Escalus', 'Friar Laurence', 'Romeo', 'Benvolio', 'Montague', 'T` with the elements in any order

Q4 Common Friends

1 Point

Given the complete Romeo and Juliet graph, what does this expression evaluate to:

```
common_friends(rj, "Romeo", "Paris")
```

- `'Mercutio'`
- `('Mercutio')`
- `['Mercutio']`
- `{'Mercutio'}`

Q5 Number Common Friends

1 Point

Given the complete Romeo and Juliet graph, what does this expression evaluate to:

```
num_common_friends_map(rj, "Paris")
```

- `['Romeo', 'Montague', 'Tybalt', 'Juliet']`
- `{'Montague': 1, 'Tybalt': 1, 'Romeo': 1, 'Juliet': 1}` with the elements in any order
- `{'Montague': 1, 'Tybalt': 1, 'Juliet': 1}` with the elements in any order
- `{'Romeo', 'Montague', 'Tybalt', 'Juliet'}` with the elements in any order
- `{'Tybalt': 2, 'Romeo': 1, 'Juliet': 1}` with the elements in any order
- `{'Montague': 2, 'Tybalt': 2, 'Romeo': 2, 'Juliet': 2}` with the elements in any order

EXPLANATION

Note that only people who have at least one friend in common with Paris will be listed. Immediate friends of Paris will not be listed (so no Mercutio, Escalus or Capulet). This is a dictionary, so order of elements is not guaranteed.

Q6 Recommend by Common Friends

1 Point

Given the complete Romeo and Juliet graph, what does this expression evaluate to:

```
recs_by_common_friends(rj, "Juliet")
```

- `{'Benvolio', 'Escalus', 'Mercutio', 'Montague', 'Paris'}` with the elements in any order
- `['Benvolio', 'Mercutio', 'Montague', 'Escalus', 'Paris']`
- `['Escalus', 'Benvolio', 'Paris', 'Mercutio', 'Montague']`
- `['Benvolio', 'Escalus', 'Mercutio', 'Paris']`
- `{'Mercutio', 'Montague', 'Benvolio', 'Montague'}` with the elements in any order
- `['Mercutio', 'Montague', 'Benvolio', 'Escalus', 'Paris']`
- `{'Mercutio', 'Paris', 'Escalus','Benvolio', 'Montague', 'Capulet'}` with the elements in any order
- (selected) `['Benvolio', 'Escalus', 'Mercutio', 'Montague', 'Paris']`
- `['Benvolio', 'Escalus', 'Mercutio', 'Montague']`

EXPLANATION

Each of these people have only one friend in common with Juliet, but the answer must be a list and it must be in sorted in alphabetical order.

Q7 Influence Map

2 Points

Given the complete Romeo and Juliet graph, what does this expression evaluate to:

```
influence_map(rj, "Paris")
```

```
{'Montague': 0.25, 'Tybalt': 0.25, 'Romeo': 0.3333333333333333, 'Juliet': 0.25 }
 in any order.
```

EXPLANATION

The friends of Paris are Capulet, Escalus and Mercutio. Thus these are the only three people who will contribute to the influence score for each person. Mercutio is the pickiest about his friends, thus he has the biggest influence (0.33). Capulet and Paris both have more friends (4 each) so their influence contribution is smaller (0.25). Only friends_of_friends of Paris are candidates for recommending as new friends for Paris, so only they will be listed in this dictionary. Romeo has only Mercutio in common with

Paris. Mercutio has three friends, so his contribution to Romeo's influence score is 0.33. Montague has only Escalus in common with Paris. Escalus has four friends, so his contribution to Montague's influence score is 0.25. Tybalt and Juliet have only Capulet in common with Paris. Since Capulet has four friends, his contribution to Tybalt and to Juliet are 0.25 each. Please work out the Mercutio example in the assert statements in the code you were given in social_network.py. That example is slightly more complex since Montague receives contributions from both Romeo (0.2) and Escalus (0.25) . Similarly, Capulet receives contributions from both Escalus (0.25) and Paris (0.33).

Q8 Recommend by Influence

2 Points

Given the complete Romeo and Juliet graph, what does this expression evaluate to:

```
recommend_by_influence(rj, "Paris")
```

```
['Romeo', 'Juliet', 'Montague', 'Tybalt']
```

EXPLANATION

This must be a list and it must consist of only these values in this exact order. Romeo comes first with the highest influence score (at 0.33), but the other three (all tied at 0.25) must be listed in alphabetical sorted order by name.