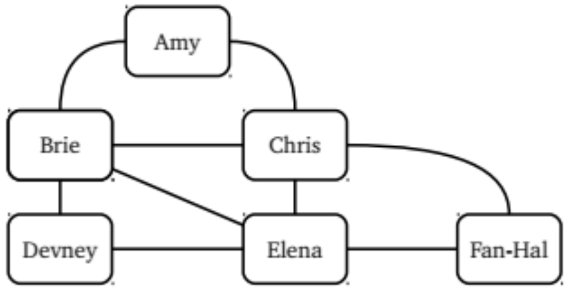
This exercise is taken from Stanford's CS106 midterm exam. Use the unit tests underneath the exercise to check your algorithm. Do not change the tests.

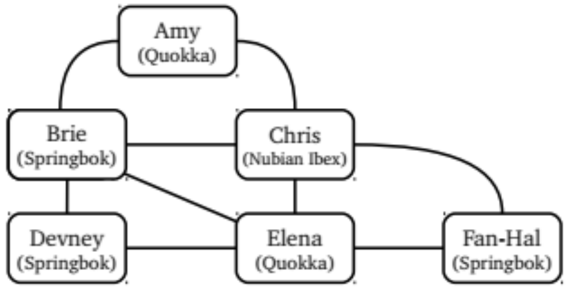
Problem Five: Animal Hipsters (24 Points)

Suppose that you have a social network represented as a graph, like this one here:



As in lecture, we will represent this graph as a `HashMap<String, ArrayList<String>>`, where each key in the `HashMap` is the name of a person and each value is an `ArrayList` of the names of the people they are friends with.

Let's suppose that every person in a social network has a favorite animal. We'll say that a person is an *animal hipster* if their favorite animal is different from all of their friends' favorite animals. For example, suppose that everyone's favorite animals are specified as follows:



Given the above social network, we would have that Amy, Chris, Elena, and Fan-Hal are animal hipsters, but Brie and Devney are not (because both of them like springboks and are they friends of one another). Even though both Amy and Elena like quokkas, they are still animal hipsters because they are not friends of one another.

Write a method

```
private ArrayList<String>
        findAnimalHipsters(HashMap<String, ArrayList<String>> network,
                           HashMap<String, String> favoriteAnimals)
```

that accepts as input a social network `network` and a `HashMap<String, String> favoriteAnimals` associating each person in the network with their favorite animal, then returns an `ArrayList<String>` containing all the people in the network who are animal hipsters.

(Continued on the next page)

In writing this method, you should assume the following:

- The `network` and `favoriteAnimals` `HashMap`s have the same set of keys, so every person in the graph has a favorite animal and everyone who has a favorite animal is in the graph.
- For simplicity, you can assume animal names are case-sensitive, so "Nubian Ibex" and "nubian ibex" should be treated as different animals.
- You are free to return the animal hipsters in any order that you'd like, though each animal hipster should appear in the list at most once.

Write your method in the space below.

```
private ArrayList<String>
findAnimalHipsters(HashMap<String, ArrayList<String>> network,
                   HashMap<String, String> favoriteAnimals) {
```

```
import static org.junit.Assert.*;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;

import org.junit.Test;

public class AnimalHipsterTest
{
	@Test
	public void testAnimalHipster()
	{
		AnimalHipster animalHipster = new AnimalHipster();

		HashMap<String, ArrayList<String>> network = new HashMap<String, ArrayList<String>>();
		HashMap<String, String> favoriteAnimals = new HashMap<String, String>();

		network.put("Amy", new ArrayList<String>(Arrays.asList(new String[] {"Brie", "Chris"})));
		network.put("Brie", new ArrayList<String>(Arrays.asList(new String[] {"Amy", "Chris", "Elena", "Devney"})));
		network.put("Chris", new ArrayList<String>(Arrays.asList(new String[] {"Amy", "Brie", "Elena", "Fan-Hal"})));
		network.put("Devney", new ArrayList<String>(Arrays.asList(new String[] {"Brie", "Elena"})));
		network.put("Elena", new ArrayList<String>(Arrays.asList(new String[] {"Brie", "Chris", "Devney", "Fan-Hal"})));
		network.put("Fan-Hal", new ArrayList<String>(Arrays.asList(new String[] {"Chris", "Elena"})));

		favoriteAnimals.put("Amy", "Quokka");
		favoriteAnimals.put("Devney", "Springbok");
		favoriteAnimals.put("Brie", "Springbok");
		favoriteAnimals.put("Chris", "Nubian Ibex");
		favoriteAnimals.put("Elena", "Quokka");
		favoriteAnimals.put("Fan-Hal", "Springbok");

		assertTrue(animalHipster.findAnimalHipsters(network, favoriteAnimals).containsAll(new
ArrayList<String>(Arrays.asList(new String[] {"Amy", "Elena", "Chris", "Fan-Hal"}))));
	}

	@Test
	public void testAnimalHipsterWithEmptyInput()
	{
		AnimalHipster animalHipster = new AnimalHipster();

		HashMap<String, ArrayList<String>> network = new HashMap<String, ArrayList<String>>();
		HashMap<String, String> favoriteAnimals = new HashMap<String, String>();

		assertTrue(animalHipster.findAnimalHipsters(network, favoriteAnimals).isEmpty());
	}

	@Test
	public void testAnimalHipsterWithEmptyNetwork()
	{
		AnimalHipster animalHipster = new AnimalHipster();

		HashMap<String, ArrayList<String>> network = new HashMap<String, ArrayList<String>>();
		HashMap<String, String> favoriteAnimals = new HashMap<String, String>();

		network.put("Amy", new ArrayList<String>());
		network.put("Brie", new ArrayList<String>());
		network.put("Chris", new ArrayList<String>());
		network.put("Devney", new ArrayList<String>());
		network.put("Elena", new ArrayList<String>());
		network.put("Fan-Hal", new ArrayList<String>());
```

```
                favoriteAnimals.put("Amy", "Quokka");
                favoriteAnimals.put("Devney", "Springbok");
                favoriteAnimals.put("Brie", "Springbok");
                favoriteAnimals.put("Chris", "Nubian Ibex");
                favoriteAnimals.put("Elena", "Quokka");
                favoriteAnimals.put("Fan-Hal", "Springbok");

                assertEquals(new HashSet<>(Arrays.asList(new String[] {"Amy", "Brie", "Chris", "Devney", "Elena", "Fan-Hal"})),
new HashSet<>(animalHipster.findAnimalHipsters(network, favoriteAnimals)));
        }
}
```