Objective. To analyze a CSV database.

This week we're going to take a closer look at the faculty of Williams. The Dean of Faculty has been kind enough to send us information on all the faculty at the college as a comma separated values (CSV) file.

We'll first build a small collection of methods to help recode the data. Then, we'll see if we can write Python to answer a number of trivia questions.

Getting the database.
As is usual, you'll need to clone the lab starter:

```
cd ~/cs134
git clone ssh://22xyz3@davey.cs.williams.edu/cs134/22xyz3/lab4.git ~/cs134/lab4
```

The resulting directory contains this lab description, a README, a CSV file that describes the faculty, two python starter files, faculty.py and trivia.py, and trivia.txt for submitting trivia answers.

Reading CSV files.
Python provides a module, csv, that will help you read (or write) CSV files. Here's how:

```
import csv
with open('faculty.csv','r') as f:
    csvf = csv.reader(f)
    for row in csvf:
        # process 'row', a list of fields separated by commas
        ...
```

The with statement says: “open this file as f only for the duration of the following suite of statements.”

You can type `pydoc3 csv` for more details.

What needs to be done.
We have two python scripts that need to be fleshed out. The first is a module, faculty, that contains a small set of tools that will help you build a database of faculty; those steps are discussed below. The second script—trivia.py—will contain routines that helped you seek answers to ten questions (attached).

First, let's write functions that will help us recode our database:

1. `parseName()`. Write a method parseName that takes a name (as a string) from our database and returns a list of names, the last of which is the surname (and possible suffix):

   ```
   >>> parseName('Nolan Jr., James L.')
   ['James', 'L.', 'Nolan Jr.]
   >>> parseName('Rulikova Edwards, Marketa')
   ['Marketa', 'Rulikova Edwards']
   ```

   We asked the Dean of Faculty for this data in Spring of 2016, and so some more recent hires have been sadly left off the list. The Computer Science Department, however, keeps its records up to date, so you'll find all of us in there!
This method is `private`; we don't expect it to be needed outside the module, so its name should not appear in `__all__`.

2. `parseDegree(s)` Write a method `parseDegree` that takes a string and returns a list that describes the degree: a year (an integer), a degree (a string), and a granting institution (a string). If the degree string is empty, return the empty list.

```python
>>> parseDegree('1988, Ph.D., University of Massachusetts, Amherst')
[1988, 'Ph.D.', 'University of Massachusetts, Amherst']
```

This definition is private, as well.

3. `parseMember(l)` Given a row (a list) read from the CSV file, return a list that describes the faculty member: item 0 is the parsed faculty name, item 1 is the faculty title, item 2 is the faculty member's department, item 3 is a parsed bachelor's degree, item 4 is the parsed master's degree, and item 5 is the parsed doctorate degree. (This routine is also private.)

```python
>>> parseMember(row) # [['Howley, Iris K.', ... ]
[['Iris', 'K.', 'Howley'],
 'Assistant Professor of Computer Science',
 'Computer Science Department',
 [2008, 'B.S.', 'Drexel University'],
 [2012, 'M.S.', 'Carnegie Mellon University'],
 [2015, 'Ph.D.', 'Carnegie Mellon University']]
```

4. `readDB(database='faculty.csv')`. Reads a database file (typically `faculty.csv`) and returns a list of parsed faculty members. This definition is used by others—it's `public`—so we include its name in `__all__`.

5. `uniq(l)` This routine takes an ordered list of values and returns a possibly smaller list with duplicate adjacent values removed. For example:

```python
```

If the list had first been sorted, the result would have been a list of unique values.

We attempted a solution, but it is broken. Please find the problem and fix it. (Apologies!)

6. `uniqCount(l)` Given a list of values, possibly containing adjacent duplicates. The `uniqCount` function returns a list of pairs, `[value,count]`. Each pair corresponds to a value that occurs in the list and the number of times it occurs, adjacently. For example:

```python
```

(Hint: This looks like run length encoding...)

When you are finished with these routines, begin answering the trivia questions on the next sheet, placing your code in `trivia.py` and your trivia responses in `trivia.txt`. Push your scripts and trivia responses by Thursday/Friday 12pm.

*
CS134C Faculty Trivia

Please write Python routines that help to answer the following questions, and submit your responses in the trivia.txt file included in the lab repository. Each routine in trivia.py should be named q1(db), q2(db), etc., where db is the result returned by faculty.readDB(). The routine should print out information that will help you accurately determine the answer. The database will be considered authoritative for the purpose of this lab.

1. How many members are on the faculty at Williams?

2. How many members of the faculty hold a doctorate?

3. What faculty member(s) hold the oldest degrees?

4. How many members of the faculty received some degree 11 years ago, in 2008?

5. How many departments are there?

6. Which department has the most faculty?

7. What is the most popular type of master's degree?

8. What is the most popular undergraduate institution?

9. What school has granted the most degrees (all forms) to Williams faculty?

10. Who has the longest isogram last name? Ignore Jr.-like suffixes, if necessary.

*