Today we will work through running examples about birthdays. These exercises will help you learn how to use dictionaries. Dictionaries were described in an earlier handout.

A **birthday month dictionary** captures information about people's birthdays. The key is a month name and the value is another dictionary! We will call these dictionaries *inner* dictionaries. For each inner dictionary, the key is the day of the month, and the value is a list of the boys and girls that were born on that day. Here's an example:

```
bdm = {'February' : {3 : ['Cat'], 15: ['Nancy'], 28: ['Al']},
       'May' : {3 : ['Katie', 'Jane'], 8 : ['Peter', 'Ed', 'Val', 'Sue']},
       'December' : {3 : ['Sharon', 'Alice'], 17 : ['Michael']}
       }
```

According to this dictionary, Cat was born on February 3rd.

Exercises

Some solutions are presented in class and also included in the moodle version of this handout.

1. Write a function that takes in a birthday month dictionary of the form described above and returns the month with the most *coverage*. The coverage of a month is the number of days within that month on which someone has a birthday. In the above example, February has higher coverage (3 days) than May or December (2 days each).

   Solution:

   ```
   def most_covered(bdm):
       '''(dict of {str:dict of {int:list of str}}) -> str
       Returns name of a month that has the most coverage.

       >>> bdm = {'December' : {24 : ['Dan', 'Joe', 'Steph']}, 'July' : {17 : ['Angelo']
       >>> most_covered(bdm)
       'July'
       '''
       max_cover = -1
       max_month = None
       for month in bdm:
           if len(bdm[month]) > max_cover:
               max_cover = len(bdm[month])
               max_month = month
       return max_month
   ```

2. Write a function that takes in a birthday month dictionary of the form described above and makes a new dictionary that contains the same information but in a different form. In the new

dictionary, the key is the person's name and the value is a list of two items, the month and day of that person's birth. For example the dictionary above would become:

```
{'Sharon': ['December', 3], 'Sue': ['May', 8], 'Val': ['May', 8],
 'Ed': ['May', 8], 'Al': ['February', 28], 'Michael': ['December', 17],
 'Alice': ['December', 3], 'Cat': ['February', 3],
 'Nancy': ['February', 15], 'Jane': ['May', 3], 'Peter': ['May', 8],
 'Katie': ['May', 3]}
```

You can assume that names are unique. Hint: use good variables to keep track of what's what.

Solution:

```
def restructure_bdm(bdm):
    '''(dict of {str:dict of {int:list of str}})
    -> dict of {str: list of str/int}
    Expects bdm to be a birthday month dictionary.
    Returns a new dictionary in which the keys are
    names of birthday boys and girls and the value
    is a list containing the month and day of that
    person's birthday.
    Expects names to be unique!
    >>> bdm = {'February' : {13 : ['Catherine']}, 'May' : {3 : ['Katie'], 8 : ['Peter
    >>> restructure_bdm(bdm)
    {'Ed': ['May', 8], 'Peter': ['May', 8], 'Catherine': ['February', 13], 'Katie': [
    '''
    name_to_bday = {}
    for month in bdm:
        day_to_names = bdm[month]
        for day in day_to_names:
            names = day_to_names[day]
            for name in names:
                name_to_bday[name] = [month, day]
    return name_to_bday

bdm = {'February' : {13 : ['Catherine']}, 'May' : {3 : ['Katie'], 8 : ['Peter', 'Ed']
print restructure_bdm(bdm)
```

3. Write a function that takes in a birthday month dictionary of the form described above and calculates which day of the month has the most birthdays aggregated over all months. On the example above, the function would return 3 because the 3^{rd} has a total of $1 + 2 + 2 = 5$ birthdays (one in February and two in May and two in December). Hint: first build a dictionary that maps each day (a number) to the *total* number of birthdays on that day aggregated across all months. For the example above, you would end up with a dictionary like this:

```
days_to_count = {3: 5, 8: 4, 15: 1, 17: 1, 28: 1}
```

Then use this dictionary to find which day has the most birthdays.

Solution:

```
def most_bdays(bdm):
    '''(dict of {str:dict of {int:list of str}}) ->  int
    Expects bdm to be a birthday month dictionary.
    Returns a the day that has the most birthdays
    aggregated over all months.
    >>> bdm = {'February' : {13 : ['Catherine']}, 'May' : {3 : ['Katie'], 8 : ['Peter
    >>> most_bdays(bdm)
    8
    '''

    # make new dictionary: day to aggregate no. of bdays
    day_to_count = {}
    for month in bdm:
        day_to_names = bdm[month]
        for day in day_to_names:
            names = day_to_names[day]
            num_bdays = len(names)
            if day not in day_to_count:
                day_to_count[day] = num_bdays
            else:
                day_to_count[day] += num_bdays
    print day_to_count
    # find day w/ largest value
    day_with_most = None
    max_bdays = 0
    for day in day_to_count:
        num_bdays = day_to_count[day]
        if num_bdays > max_bdays:
            max_bdays = num_bdays
            day_with_most = day
    return day_with_most



bdm = {'February' : {3 : ['Cat'], 15: ['Nancy'], 28: ['Al']},
       'May' : {3 : ['Katie', 'Jane'], 8 : ['Peter', 'Ed', 'Val', 'Sue']},
       'December' : {3 : ['Sharon', 'Alice'], 17 : ['Michael']}
       }
```

```
print most_bdays(bdm)
```

The birthday month problem is adapted from Zingaro.