CSE 160 Section 7 Problems

1. Write code that, given a list of dictionaries, creates a single dictionary containing the sums of values with the same key in the given dictionaries. For example:
Given this list of dictionaries:

```
[{'b': 10, 'a': 5, 'c': 90},
  {'b': 78, 'a': 45},
  {'a': 90, 'c': 10}]
```

Your code should create : `{'b': 88, 'a': 140, 'c': 100}`

2. Write a function freq that takes a string as an argument, and returns a dictionary that maps each character to its frequency in the given string. For example, freq("Star Wars") should return: {"S":1, "t":1, "a":2, "r":2, " ":1, "W":1, "s":1}.

3. What output is produced after running the following piece of code?

```
from operator import itemgetter
data = [ ("Fred", 3, 5),
          ("Zeke", 5, 3),
          ("Sam", 5, 6),
          ("Mary", 3, 5),
          ("Ann", 7, 8) ]
def some_key(x):
    return len(x[0])
print(sorted(data, key=some_key))
print(sorted(data, key=itemgetter(2), reverse=True))
```

4. Write a function get_youngest_person that takes a list of dictionaries as an argument and returns the name of the youngest person in the list. The list of dictionaries will have the following format:

```
people = [ {"name": "Alice", "age": 20},

           {"name": "Bob", "age": 9},

           {"name": "Dan", "age": 56}]
```

For example, `get_youngest_person(people)` should return `"Bob"`. If there is more than one person with the smallest age, return the name of the person who occurs first in the list. You may assume the list contains at least one person and that no one is less than 1 year old.

5. Write a function `word_lengths` that takes a string argument. Assume the string has already been stripped of all punctuation and converted to lowercase. The function should split the string into individual words and return a dictionary mapping the number of letters in a word to a set of words of that length that appeared in the string. For example, calling:

```
print(word_lengths("this is a cool string eh"))
```

Would print something like this:

```
{1: set(['a']), 4: set(['this', 'cool']), 2: set(['is', 'eh']),
       6: set(['string'])}
```

6. Write a function `reverse_dict(to_reverse)` that returns a NEW dictionary that is the reverse of the dictionary (reversed key and value) `to_reverse`. Assume `to_reverse` will have unique keys and values.

For example, if `to_reverse` is: `{ 'a': 1, 'b': 2, 'c': 3 }`

Then this function should return `{ 1: 'a', 2: 'b', 3: 'c' }`