CSE 160 Section 7 Problems

1. Write a function called `all_unique_words(file_name)` that takes in a string `file_name` and returns the number of unique words in the file. You may use the `split()` function for this problem, which takes in a string and returns a list of the words in the string separated by empty spaces.
   Example:
   If colors.txt has the content "`red green blue green`"
   Your output should be: 3

```
def all_unique_words(file_name):
     file = open(file_name):
     words = file.read()
     unique = set(words.split())
     file.close()
     return len(unique)
```

2. 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))
```

[('Sam', 5, 6), ('Ann', 7, 8), ('Fred', 3, 5), ('Zeke', 5, 3), ('Mary', 3, 5)]
[('Ann', 7, 8), ('Sam', 5, 6), ('Fred', 3, 5), ('Mary', 3, 5), ('Zeke', 5, 3)]

3.

a. Given a list of tuples in the form `(name, age)`, using itemgetter, return a list of names and ages sorted alphabetically in one line of code

```
alphabetical_lst = sorted(lst, key = itemgetter(0))
```

b. Given a list of tuples in the form `(name, age)`, using itemgetter, return a list of names and ages sorted by increasing age in one line of code

```
youngest_to_oldest = sorted(lst, key = itemgetter(1))
```

c. Using `itemgetter`, define a function called `find_oldest` that takes in a list of tuples in the form `(name, age)` and returns a list of tuples belonging to the oldest people. If there is a tie, return a list of the names and ages of the people sharing the same (oldest) age in a new list in alphabetical order. For example, given `age_list = [ ("Tom", 19), ("Max", 26), ("James", 12), ("Alice", 26), ("Carol", 10) ]`, `find_oldest(age_list)` would return `[("Alice", 26), ("Max", 26)]`

```
def find_oldest(age_list):
```

```
def find_oldest(age_list):
    sort_name = sorted(age_list, key=itemgetter(0))
    sort_age = sorted(sort_name, key=itemgetter(1), reverse=True)
    oldest_age = sort_age[0][1]
    ret_list = []
    for pair in sort_age:
        if (pair[1] == oldest_age):
            ret_list.append(pair)
        else:
            return ret_list # return early since it is sorted
    return ret_list
```

4. You are given a list of dictionaries representing the scientific and common names of various plants. An example is shown below:

```
cactus = [{“scientific name”: “Kroenleinia grusonii”, ”common
          name”: “golden barrel cactus”},
          {“scientific name”: “Kroenleinia grusonii”, ”common
          name”: “golden ball”},
          {“scientific name”: “Carnegiea gigantea”, ”common
          name”: “saguaro”}]
```

Write a function called `unique_species(plants)` that takes in a list of dictionaries and returns a list of all of the unique species by their scientific name sorted alphabetically (hint: use sets!). In the above the example this function should return:

```
[“Carnegiea gigantea”, “Kroenleinia grusonii”]
```

```
def unique_species(plants)
```

```
def unique_species(plants):
    species = set()
    for name in cactus:
        species.add(name["scientific name"])
    return sorted(list(species))
```