CSE 160 Section 7 Problems

Sets are data types that contain a group of unordered, unique, and immutable values. Let's create some sets:

```
my_set = set()   # empty set
my_set = set{1,2,3}   # set initialized with values 1,2,3
my_set = {[1,2,3]} # set initialized from a list
```

Below are some useful set operations, try them out to better understand what they do!

Set Operation	Code
Adding values	`my_set.add(val)`

Removing a value	`my_set.remove(val) #Value must already exist`
	`my_set.discard(val) #Value doesn't need to exist`

Return a random element	`my_set.pop()`
Return all values in both sets	`set_1 | set_2`
Return values found in both sets	`set_1 & set_2`
Return values only found in set_1	`set_1 - set_2`
Return values not found in both sets	`set_1 ^ set_2`

Tuples are data types that are ordered and unchangeable. Let's create some tuples:

```
my_tuple = tuple(1,)   # tuple with one element
my_tuple = tuple(1,2,3)   # tuple initialized with values 1,2,3
my_tuple = ([1,2,3]) # tuple initialized from a list
```

Tuples share many similarities with lists in indexing, slicing, and looping. However, below are some key differences between the two

- Tuples cannot be changed or appended to, but lists can
- Tuples can be put into a set and used as the key of a dictionary, which lists cannot

---

1. After the following lines of code are executed, what values are stored in the set `output_set`?

```
input_list = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 9]
output_set = set()
for i in input_list:
     output_set.add(i)
```

2. Given:

```
set_one = {'a', 'b', 'c', 'd', 'e', 'f'}
set_two = {'a', 'c', 'd', 'g'}
```

In one line of code, print the set of all letters that are:
(a) in both the sets. (i.e. - their intersection)

(b) in either of the sets. (i.e. - their union)

(c) in `set_one` but not in `set_two`

(d) in `set_two` but not in `set_one`

3. Write a function called `all_unique_words` 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 space.

Example:
If colors.txt has the content “red green blue green”
print(all_unique_words(“colors.txt”)) outputs 3

```
def all_unique_words(file_name):
```

4. 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))
```

5.

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

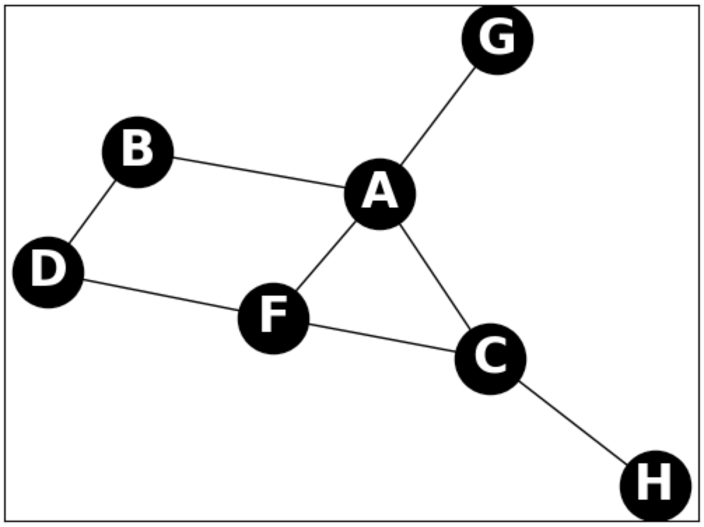
b. Given a list of tuples in the form `(name, age)`, using itemgetter, return a list names and ages sorted alphabetically in one line of code

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):
```

6. Define a function called `two_stops_away(graph, stop)` that, when given a graph of bus stops and the name of one bus stop, returns a set of bus stop names that are two stops away from the given bus stop. **You should not include the given bus stop in your result.** You can assume that the `stop` exists in the `graph`, and there exists a helper function, `get_next_stops(graph, stop)` that takes in a graph and a given bus stop name and returns a set of bus stop names that can be reached directly from that bus stop.

For example, given the following graph of bus stops (stored in a variable `graph`):



`two_stops_away(graph, 'F')` should return the set `{'A', 'C', 'B', 'G', 'H'}`. Note that even though we can go to A directly from F, we include it because we can also go to A via C. Same goes for C. If we call the helper function `get_next_stops(graph, 'F')`, it returns the set `{'A', 'C', 'D'}`.

7. The social media company Yipper has asked you to help debug some code they've been having issues with. They've given you the following exception:

```
Traceback (most recent call last):
  File "yipper.py", line 21, in <module>
    likes = likes_per_user(yips)
  File "yipper.py", line 14, in likes_per_user
    result[i][1] = result[i][1] + yip['likes']
TypeError: 'tuple' object does not support item assignment
```

a. What does this exception mean? What has gone wrong?

b. The company has given you the following code. How could it be fixed to avoid this error?

```
def get_yips():
    return [
        {'name': 'Chewbarka', 'likes': 16,
         'message': 'Met a snake yesterday; seemed pretty cool \
                     #python'},
        {'name': 'Mary Puppins', 'likes': 0,
         'message': 'Today was pretty exceptional! \
                     #notavalueerror'},
        {'name': 'Chewbarka', 'likes': 3,
         'message': 'Started playing a new game today! #anemo'}
    ]

def likes_per_user(yips):
    result = []
    for yip in yips:
        found = False
        for i in range(len(result)):
            if result[i][0] == yip['name']:
                result[i][1] = result[i][1] + yip['likes']
                found = true
        if not found:
            result.append((yip['name'], yip['likes']))
    return result

yips = get_yips()
likes = likes_per_user(yips)
print(likes)
```