CSE143X Section #9 Problems

For all problems involving maps, the contents will be displayed using the standard key=value pairs enclosed in curly braces used by toString. For example, given the following Map:

```
        Map<Integer, String> months = new TreeMap<>();
        months.put(3, "March");
        months.put(1, "January");
        months.put(2, "February");
```

the map would be displayed as follows:

```
        {1=January, 2=February, 3=March}
```

Map<K, V> Methods (11.3)

put(key, value)	adds a mapping from the given key to the given value
get(key)	returns the value mapped to the given key (null if none)
containsKey(key)	returns true if the map contains a mapping for the given key
remove(key)	removes any existing mapping for the given key
clear()	removes all key/value pairs from the map
size()	returns the number of key/value pairs in the map
isEmpty()	returns true if the map's size is 0
keySet()	returns a Set of all keys in the map
values()	returns a Collection of all values in the map
putAll(map)	adds all key/value pairs from the given map to this map

1. Consider the following method:

```
    public Set<String> mystery(Map<String, String> data) {
        Set<String> result = new TreeSet<>();
        for (String s : data.keySet()) {
            result.add(data.get(s));
        }
        return result;
    }
```

The three entries below have specific values for the parameter to method mystery. For each entry, indicate what values would be stored in the set returned by method mystery if the given maps are passed as parameters.

map: {baz=c, mumble=d, foo=a, bar=b}

set returned:____

map: {f=z, d=x, e=y, b=y, c=z, a=x}

set returned:____

map: {f=2, g=10, d=20, e=1, b=10, c=2, a=1, h=20}

set returned:____

2. Write a method counts that accepts a List of integers and a Set of integers as parameters, and returns a map from each value in the set to the number of occurrences of that value in the list. For example, if your method is passed the following list and set as parameters:

```
list: [4, -2, 3, 9, 4, 17, 5, 29, 14, 87, 4, -2, 100]
set:  [-2, 4, 29]
```

   Then your method should return the map {-2=2, 4=3, 29=1}, because there are two occurrences of -2, three occurrences of 4, and one occurrence of 29.

3. Write a method called split that takes a set of strings as a parameter and that returns the result of splitting the strings into different sets based on the length of the strings. In particular, your method should return a map whose keys are integers and whose values are sets of strings of that length. For example, if a variable called words contains the following set of strings:

```
[to, be, or, not, that, is, the, question]
```

   then the call split(words) should return a map whose values are sets of strings of equal length and whose keys are the string lengths:

```
{2=[be, is, or, to], 3=[not, the], 4=[that], 8=[question]}
```

   Notice that strings of length 2 like "be" and "is" appear in a set whose key is 2. If the set had instead stored these strings:

```
[four, score, and, seven, years, ago, our, fathers, brought, forth]
```

   Then the method would return this map:

```
{3=[ago, and, our], 4=[four], 5=[forth, score, seven, years],
 7=[brought, fathers]}
```

4. Write a method reverse that accepts a Map from integers to strings as a parameter and that returns a new Map of strings to integers that is the original's "reverse". The reverse of a map is defined here to be a new map that uses the values from the original as its keys and the keys from the original as its values. Since a map's values need not be unique but its keys must be, it is acceptable to have any of the original keys as the value in the result. In other words, if the original map has pairs (k1, v) and (k2, v), the new map must contain either the pair (v, k1) or (v, k2).

   For example, for the following map:

```
{42=Marty, 81=Sue, 17=Ed, 31=Dave, 56=Ed, 3=Marty, 29=Ed}
```

   Your method could return the following new Map:

```
{Dave=31, Ed=29, Marty=3, Sue=81}
```

   The keys of the new Map should be sorted alphabetically.

5. Write a method maxOccurrences that accepts a List of integers as a parameter and that returns the number of times the most frequently occurring integer (the "mode") occurs in the list. Solve this problem using a single Map as auxiliary storage. If the list is empty, return 0.

6. Write a method called convert that takes as a parameter a set of strings representing phone numbers and that returns a map of strings to sets of strings that represent the same phone numbers split into exchange/suffix. In particular, the phone numbers in the set passed to the method will each include a 3-digit exchange followed by a dash followed by a four-digit suffix, as in:

```
[493-3923, 723-9278, 384-1917, 555-1795, 384-4923, 555-4923, 555-1212,
 723-9823]
```

   The method should split each string into the 3-digit exchanges that come before the dash and the suffixes that come after. The method should construct a map in which the keys are the 3-digit exchanges. Each such exchange will be mapped to a set of 4-digit suffixes. For the set of phone numbers above, the following map would be constructed:

```
{384=[1917, 4923], 493=[3923], 555=[1212, 1795, 4923], 723=[9278, 9823]}
```

   Notice, for example, that three of the phone numbers in the original set began with "555". In the map, the key "555" maps to a set of three elements (the three suffixes that came after "555-").

   Your method should construct the new map and each of the sets contained in the map. Recall that the String class has a substring method that takes a starting index (inclusive) and a stopping index (exclusive). For example:

```
"Australia".substring(1, 5) returns "ustr"
```

   The keys of the new map should be ordered by the 3-digit exchanges and each set should be ordered by the 4-digit suffixes in that set.

7. Write a method called acronyms that takes a set of word lists as a parameter and that returns a map whose keys are acronyms and whose values are the word lists that produce that acronym. Acronyms are formed from each list as described in problem 1. Recall that the list [laughing, out, loud] produces the acronym "LOL". The list [League, of, Legends] also produces the acronym "LOL". Suppose that a variable called lists stores this set of word lists:

```
[[attention, deficit], [Star, Trek, Next, Generation],
 [laughing, out, loud], [International, Business, Machines],
 [League, of, Legends], [anno, domini], [art, director],
 [Computer, Science and, Engineering]]
```

   Each element of this set is a list of values of type String. You may assume that each list is nonempty and that each string in a list is nonempty.

   Your method should construct a map whose keys are acronyms and whose values are sets of the word lists that produce that acronym. For example, the call acronyms(lists) should produce the following map:

```
{AD=[[attention, deficit], [anno, domini], [art, director]],
 CSE=[[Computer, Science and, Engineering]],
 IBM=[[International, Business, Machines]],
 LOL=[[laughing, out, loud], [League, of, Legends]],
 STNG=[[Star, Trek, Next, Generation]]}
```

Notice that there are 5 unique acronyms produced by the 8 lists in the set. Each acronym maps to a set of the word lists for that acronym. Your method should not make copies of the word lists; the sets it constructs should store references to those lists. As in the example above, the keys of the map that you construct should be in sorted order. You may assume that a method called acronymFor is available that takes a list of strings as a parameter and that returns the corresponding acronym. Your method is not allowed to change either the set passed as a parameter or the lists within the set. The sets that you construct for the new map will have to be of type HashSet (we will explore why later in the course).

8. Write a method called deepCopy that takes as a parameter a map whose keys are strings and whose values are lists of integers and that creates and returns a new map that is a copy of the map parameter. For example, given a variable called map that stores the following information:

```
{"cse143"=[42, 17, 42, 42], "goodbye"=[3, 10, -5],
 "hello"=[16, 8, 0, 0, 106]}
```

   the call deepCopy(map) should return a new map whose structure and content are identical to map. Any later modifications to map or the lists in map following this call should not be reflected in the copy. The map you construct should store keys in alphabetical order. Your method should not modify the contents of the map passed as a parameter. In constructing collection objects, you are required to use the 0-argument constructors.

9. Write a method called extractEqual that takes a set of Point objects and that returns a new set that contains all of the Point objects where the x and y values are equal to each other. For example, if a set called points contains the following values:

```
[[x=42,y=3], [x=4,y=2], [x=18,y=1], [x=7,y=8], [x=-2,y=-2], [x=3,y=3],
 [x=7,y=7], [x=0,y=82], [x=14,y=14], [x=3,y=13], [x=-3,y=4], [x=1,y=3]]
```

   then the call extractEqual(points) should return the following set:

```
[[x=-2,y=-2], [x=3,y=3], [x=7,y=7], [x=14,y=14]]
```

   The original set should be unchanged and you should not construct any new Point objects in solving this problem. As a result, both sets will end up referring to the Point objects in which the x and y coordinates are equal. Your method is expected to have reasonable efficiency in that it shouldn't lead to more set operations than it needs to. The set that you construct to return will have to be of type HashSet (we will explore why later in the course).