Problem 1: Sets
Login to Practice it and solve the problem provided in this link
https://practiceit.cs.washington.edu/problem/view/cs2/exams/midterms/midterm6/commonFirstLetters

Problem2: UnSortedTableMap

Given the UnsortedTableMap1.java class which implements Map interface which binds a key uniquely to a value. After reviewing and understanding the code provided for UnSortedTable1.java, implement a new Boolean method isSubMap(UnsortedTableMap1 map2) which accepts an UnsortedTableMap as a parameter and returns true if every key in the first map (table) is also contained in the second map (map2) and maps to the same value in the second map. Consider, the following maps:
map1: {Smith=949-0504, Marty=206-9024}
map2: {Marty=206-9024, Hawking=123-4567, Smith=949-0504, Newton=123-4567}
map3: {Alisha=321-7654, Hawking=123-4567, Smith=888-8888}
map4: {}
Given the maps above, map1 is a sub-map of map2, so the call of map1.isSubMap(map2) would return true. The order of the parameters does matter, so the call of map2.isSubMap(map1) would return false.
But map3 is not a sub-map of map2 because the key "Alisha" is not in map2 and also because the key "Smith" does not map to the same value as it does in map2; therefore, the call of map3.isSubMap(map2) would return false. The empty map is considered to be a sub-map of every map, so the call of map4.isSubMap(map1) would return true.

Tester

```
UnsortedTableMap1<String, String> map1 = new UnsortedTableMap1<>();
map1.put("Smith", "949-0504");
map1.put("Marty", "206-9024");
UnsortedTableMap1<String, String> map2 = new UnsortedTableMap1<>();
map2.put("Marty","206-9024");
map2.put("Hawking","123-4567");
map2.put("Smith","949-0504");
map2.put("Newton","123-4567");
UnsortedTableMap1<String, String> map3 = new UnsortedTableMap1<>();
map3.put("Alisha","321-7654");
map3.put("Hawking","123-4567");
map3.put("Smith","888-8888");
UnsortedTableMap1<String, String> map4 = new UnsortedTableMap1<>();
System.out.println(map1.isSubMap(map2));
System.out.println(map2.isSubMap(map1));
System.out.println(map1.isSubMap(map3));
System.out.println(map4.isSubMap(map1));
```

Output

```
true
false
false
true
```

Problem 3: Maps

In this exercise, you are given a collection of documents (a data set). We provide the data set reuters.zip for experimenting. The dataset contains 2,264 news articles from Reuters. Each article is assigned to one of the following five categories: acq, earn, crude, interest, and trade. You can determine the category of an article simply by looking at the suffix of its file name. You must read all the words from all the documents and store them in appropriate data structures (Lists, Maps, Sets, …) to respond to the following queries:

1. containsD(String term, String document): returns true if term exists in the given document.
2. containsC(String term, String category): returns true if term exists in any of the files of the given category.
3. countD(String term, String document): returns how many times the term occurs in the given document.
4. countC(String term, String category): returns how many times the term occurs in the all the documents of the given category.
5. maxD(String document): returns the term that occurred the max number of times in the given document.
6. maxC(String category): returns the term that occurred the max number of times in all the documents of the given category.
7. similarity(String doc1, String doc2): returns the similarity score between the two given documents. The similarity is the ratio of the number of common words in doc1 and doc2 to the total number of unique words in both doc1 and doc2, as given by the following formula:

$$sim(doc1, doc2) = \frac{|doc_1 \cap doc_2|}{|doc_1 \cup doc_2|}$$

You are provided with the starter code of the class Reuters.java. The class contains two attributes mapD and mapC. You are free to use only one map implementation, for this exercise, rather than two.

Problem4: Open Addressing and Linear Probing

1) In linear probing technique, collision is resolved by searching linearly in the hash table until an empty location is found.

The keys 12, 18, 13, 2, 3, 23, 5 and 15 are inserted into an initially empty hash table of length 10 using open addressing with hash function h(k) = k mod 10 and linear probing. What is the resultant hash table?

0	
1	
2	2
3	23
4	
5	15
6	
7	
8	18
9	

(A)

0	
1	
2	12
3	13
4	
5	5
6	
7	
8	18
9	

(B)

0	
1	
2	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

(C)

0	
1	
2	12, 2
3	13, 3, 23
4	
5	5, 15
6	
7	
8	18
9	

(D)

2) A hash table of length 10 uses open addressing with hash function h(k)=k mod 10, and linear probing. After inserting 6 values into an empty hash table, the table is as shown below.

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

Which one of the following choices gives a possible order in which the key values could have been inserted in the table?
(A) 46, 42, 34, 52, 23, 33
(B) 34, 42, 23, 52, 33, 46
(C) 46, 34, 42, 23, 52, 33
(D) 42, 46, 33, 23, 34, 52

3) How many different insertion sequences of the key values using the same hash function and linear probing will result in the hash table given in Question C above?

(A) 10
(B) 20
(C) 30
(D) 40

4) Consider a hash table with 100 slots. Collisions are resolved using chaining. Assuming simple uniform hashing, what is the probability that the first 3 slots are unfilled after the first 3 insertions? (GATE-CS-2014)
(A) (97 × 97 × 97)/100^3
(B) (99 × 98 × 97)/100^3
(C) (97 × 96 × 95)/100^3

5) Which one of the following hash functions on integers will distribute keys most uniformly over 10 buckets numbered 0 to 9 for i ranging from 0 to 2020?
(A) h(i) =i^2 mod 10
(B) h(i) =i^3 mod 10
(C) h(i) = (11 ∗ i^2) mod 10
(D) h(i) = (12 ∗ i) mod 10