This lab includes a few programming exercises to make sure that you know how to write algorithms using sets and maps from the Java Collections Framework.

Deliverables and Grading:

- submit your **Java code** and **print screen of output** for each question to *Lab7* on D2L
- Please do not zip or compress your submissions. D2L allows you to upload multiple files

1. [2 mark] Write a program that uses Java Map to count the number of times each word appears in the Beatle's song "All You Need is Love". Test your program using the lyrics, which are available in file love.txt on D2L.

   The output from your program should be sorted in alphabetical order by song word, and should look something like this:

   ```
   all - 6
   big - 8
  love - 4

   ... etc
   ```

   Note that punctuation has already been removed from the input file, but it is still mixed case and has whitespace. Implement your algorithm in java, test it, show the instructor, and submit it to D2l.

2. [3 mark]Write a program that ***uses a HashSet*** to determine if all the words in a file are distinct (ie: different). Your program should print `DISTINCT` or `NOT DISTINCT` based on the contents of the file. Test your program with q2input.txt.

3. [5mark] for this question test your solution with the input file q3Test.txt which is available on D2L.
A BCIT instructor is having a problem with students giving fictitious excuses when they are late with their homework. In order to reduce the amount of time spent listening to fictitious excuses, the instructor has asked you to write a program that will search a list of excuses for keywords to help identify the fictitious excuses.

Input

Input is from a file, with a single test case per file. The first line contains exactly two integers: K (which defines the number of keywords), followed by E (which defines the number of excuses).
Lines 2 through K+1 each contain exactly one keyword, and lines *K*+2 through *K*+1+*E* each contain exactly one excuse.
Keywords will be formed of contiguous lower case alphanumeric characters (ie: no white space or punctuation). Excuses are strings of blank delimited lower case words. The words in the excuses contain only lower case alphanumeric characters.

Output

Your algorithm will print the *worst* excuse(s), one per line, formatted exactly as read in. We define the *worst excuse(s)* as the excuse(s) which contains the largest number of keywords.

If a keyword occurs more than once in an excuse, each occurrence of the keyword is counted. A keyword "occurs" in an excuse if and only if it exists in the string in contiguous form and is delimited by the beginning or end of the line or a space.
Note: If there is more than one worst excuse, they must be printed in lexicographical order. Duplicate excuses are only printed once.

Sample Input

```
4 5
postdog
ate
homework
canary
this excuse is so good that it contain no keywords
a postdog ate my homework
the canary ate my postdog
my canary was killed by a postdog
can you believe postdog died
```

Sample Output

```
a postdog ate my homework

the canary ate my postdog
```