Since the lyrics of most pop music contains words that repeat a number of times, a simple way to compress a lyric file is to create a map that stores each word once along with the positions of each word in the file.

For example, suppose the lyric consists of the lines:

What have I

1 2 3 <-- word position in lyrics

What have I

4 5 6 <-- word position in lyrics

What have I done to deserve this

7 8 9 10 11 12 13 <-- word position in lyrics

We would form a map that maps each unique word to a list of word positions in the lyric. NOTE: The word position for a word at the end of a line is stored as a negative integer rather than a positive integer so you can recreate the lyrics later when you iterate through the words in the map.

Sample map for the lyric above (order of words may vary):

Word Word Position(s)

===========================

WHAT 1, 4, 7

HAVE 2, 5, 8

I -3, -6, 9

DONE 10

TO 11

DESERVE 12

THIS -13

**Assignment**

This project currently contains only one class that models the analyzer using a HashMap that maps a lyric word to a list of integers representing the locations of the word in the lyrics.

1. Complete the class for the lyric analyzer by adding the following methods listed below.
   1. Write an add method with the following signature:

public void add(HashMap<String,ArrayList<Integer>> map, String lyricWord, int wordPosition)

This method will determine if the given lyric word is in the map. If the word is not in the map, then a mapping is added that maps that word to a list containing the position of the word in the lyric. If the word is already in the map, then its word position is added to the list of word positions for this word. Do not create a new mapping if the lyric word is already in the map. Use the one that is already there and just update its list. Remember to negate the word position if the word is at the end of a line in the lyrics.

* 1. Write a displayWords method with the following signature:

public void displayWords(HashMap<String,ArrayList<Integer>> map)

This method should display the words of the song along with the word positions of the word, one word per line, **in alphabetical order**. You should do this **without creating another map**. Instead, get a set of all the words stored in the map. Sort this set using one of the sort methods from the Java API. Then iterate over this sorted set and print out each word along with the word positions associated with each word. You may leave the negative integers in the word position list. (see sample output below) **Iterate through the array of words using the for loop.**

* 1. Write a displayLyrics method with the following signature:

public void displayLyrics(HashMap<String,ArrayList<Integer>> map)

This method will display the lyrics for the song (in uppercase) stored in the map. Start with an empty array of strings whose size is the number of words in the lyric plus 1 (don't use cell 0). Initialize this array with empty strings (not null). Then, get a set of all of the words stored in the map. For each word, store the word in the array cells corresponding to its word position(s). If a word position is negative, add on an extra newline character to the word when you store the word in the array. Once you finish processing all words that are in the map, iterate through the array and print out each string, and you should see the lyrics appear, line by line. **Iterate through the array of words using the for loop.**

* 1. Write a method with the following signature:

public int count(HashMap<String,ArrayList<Integer>> map)

This method will return the total number of **unique** words in the lyric by analyzing the map.

* 1. Write a method with the following signature:

public String mostFrequentWord(HashMap<String,ArrayList<Integer>> map)

This method will return the word that occurs most frequently in the lyric. Do this by getting a set of all the words in the map and then for each word, find the one that has the largest set of word positions. Do not create any additional data structures to solve this problem. If there is a tie for the most frequent word, print out the most frequent word that comes first alphabetically.

1. Use the main method to test your methods. Don’t forget to change the file path.

**Testing/Sample Output**

The test1.txt output for the simple lyric given above would look like this:

WHAT HAVE I

WHAT HAVE I

WHAT HAVE I DONE TO DESERVE THIS

DESERVE: 12

DONE: 10

HAVE: 2,5,8

I: -3,-6,9

THIS: -13

TO: 11

WHAT: 1,4,7

The number of unique words in the lyric is: 7

Most frequent word: HAVE