```
public class HashTagTokenizer {

    public static void main(String[] args) {

        String hashTag = args[0];
        String []dictionary = readDictionary("dictionary.txt");
        breakHashTag(hashTag, dictionary);


    }

    public static String[] readDictionary(String fileName) {
        String[] dictionary = new String[3000];

        In in = new In(fileName);
        int i = 0;

        while (!in.isEmpty()){

            dictionary[i] = in.readLine();
            i++;

        }

        return dictionary;
    }

    public static boolean existInDictionary(String word, String []dictionary) {

        for (int i = 0; i < dictionary.length; i++){

            if (word.equals(dictionary[i])){

                return true;
            }
        }

        return false;
    }
```

```
public static void breakHashTag(String hashtag, String[] dictionary) {

    hashtag = hashtag.toLowerCase();

    // Base case: do nothing (return) if hashtag is an empty string.
    if (hashtag.isEmpty()) {
        return;
    }

    int N = hashtag.length();

    for (int i = 1; i <= N; i++) {

        if(existInDictionary(hashtag.substring(0, i), dictionary)){

            System.out.println(hashtag.substring(0, i));
            breakHashTag(hashtag.substring(i, N), dictionary);

            }
        }


    }
}
```

```
public class SpellChecker {


    public static void main(String[] args) {

        String word = args[0];
        int threshold = Integer.parseInt(args[1]);
        String[] dictionary = readDictionary("dictionary.txt");
        String correction = spellChecker(word, threshold, dictionary);
        System.out.println(correction);

    }

    public static String tail(String str) {

        String str1 = "";

        if(str.length() == 1) return str1;

        else return str1 = str.substring(1);

    }

    public static int levenshtein(String word1, String word2) {

        word1 = word1.toLowerCase();
        word2 = word2.toLowerCase();

        int num;

        if(word1.isEmpty()) return word2.length();

        if(word2.isEmpty()) return word1.length();

        char headA = word1.charAt(0);
        char headB = word2.charAt(0);
        String tailA = tail(word1);
        String tailB = tail(word2);

        if(headA == headB) return num = levenshtein(tailA, tailB);
```

```
    else {

        int min1 = Math.min(levenshtein(tailA, word2), levenshtein(word1, tailB));
        num = 1 + Math.min(min1, levenshtein(tailA, tailB));
    }

    return num;
}

public static String[] readDictionary(String fileName) {
    String[] dictionary = new String[3000];

    In in = new In(fileName);
    int i = 0;

    while (!in.isEmpty()){

        dictionary[i] = in.readLine();
        i++;

    }

    return dictionary;
}

public static String spellChecker(String word, int threshold, String[] dictionary) {
    int lev;
    int[] arr = new int[3000];
    int minIndex = 0;


    for (int i = 0; i < 3000; i++){

        lev = levenshtein(word, dictionary[i]);
        arr[i] = lev;
    }

    int min = arr[0];
```

```
        for (int i = 0; i < arr.length; i++){

            if(arr[i] < min)  {

                min = arr[i];
                minIndex = i;


            }
        }

        String str = dictionary[minIndex];

        if (min > threshold) return word;
        else return str;

    }

}
```