```
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);
        for (int i = 0; i < dictionary.length; i++) {
            dictionary[i] = in.readString();
        }
        return dictionary;
    }
    public static boolean existInDictionary(String word, String []dictionary) {
        for (int i = 0; i < dictionary.length; i++) {
            if(dictionary[i].equals(word)){
                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++) {
            String pixel = hashtag.substring(0, i);
            if (existInDictionary(pixel, dictionary)) {
                System.out.println(pixel);
                breakHashTag(hashtag.substring(i), dictionary);
            }
        }
    }
}
```

```
public class SpellChecker {
    public static void main(String[] args) {
        //dont change
        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) {
        if(str.length() == 1){
            return "";
        }
        return str.substring(1);
    }
    public static int levenshtein(String word1, String word2) {
        word1 = word1.toLowerCase();
        word2 = word2.toLowerCase();
        if (word2.length() == 0) {
            return word1.length();
        }
        if (word1.length() == 0) {
            return word2.length();
        }
        String head1 = word1.substring(0,1);
        String head2 = word2.substring(0,1);
         if (head1.equals(head2)) {
            return levenshtein(tail(word1), tail(word2));
        }
        else {
            int a = levenshtein(tail(word1), word2);
            int b = levenshtein(word1, tail(word2));
            int c = levenshtein(tail(word1), tail(word2));
            return 1 + Math.min(a, Math.min(b, c));
        }
    }
    public static String[] readDictionary(String fileName) {
        String[] dictionary = new String[3000];
        In in = new In(fileName);
        for (int i = 0; i < dictionary.length; i++) {
            dictionary[i] = in.readString();
        }
        return dictionary;
    }
```

```
    public static String spellChecker(String word, int threshold, String[] dictionary) {
        int min = 1 + threshold;
        String closestWord = word;
        for (int i = 0; i < dictionary.length; i++) {
            String dictWord = dictionary[i];
            int distance = levenshtein(word, dictWord);
            //the calculated distance is less than the current min update min
            if (distance < min) {
                min = distance;
                closestWord = dictWord;
            }
        }
        if (min <= threshold) {
            return closestWord;
        } else {
            return word;
        }
    }
}
```