public class HashTagTokenizer {

public static void main(String[] args) {

String hashTag = args[0];

String[] dictionary = readDictionary("dictionary.txt");

breakHashTag(hashTag.toLowerCase(), dictionary);

}

public static String[] readDictionary(String fileName) {

String[] dictionary = new String[3000];

In file = new In(fileName);

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

dictionary[i] = file.readLine();

}

file.close();

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) {

if (hashTag.isEmpty()) {

return;

}

int N = hashTag.length();

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

if (existInDictionary(hashTag.substring(0,i), dictionary) == true){

System.out.println(hashTag.substring(0,i));

breakHashTag(hashTag.substring(i), dictionary);

}

}

}

}

public class SpellChecker {

public static void main(String[] args) {

String word1 = args[0];

int threshold = Integer.parseInt(args[1]);

String[] dictionary = readDictionary("dictionary.txt");

String correction = spellChecker(word1, threshold, dictionary);

System.out.println(correction);

}

public static String head(String str) {

if (str.isEmpty()){

return "";

}

return str.substring(0, 1);

}

public static String tail(String str) {

if (str.length() == 1){

return "";

}

return str.substring(1);

}

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

if (word1.length() > word2.length()) {

return levenshtein(word1.substring(0, word2.length()), word2) + (word1.length() - word2.length());

}

else if (word1.length() < word2.length()) {

return levenshtein(word1, word2.substring(0, word1.length())) + (word2.length() - word1.length());

}

else if (word1.isEmpty()) {

return 0;

}

else {

if (head(word1.toLowerCase()).equals(head(word2.toLowerCase()))) {

return levenshtein(tail(word1), tail(word2));

}

return levenshtein(tail(word1), tail(word2)) + 1;

}

}

public static String[] readDictionary(String fileName) {

String[] dictionary = new String[3000];

In file = new In(fileName);

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

dictionary[i] = file.readLine();

}

file.close();

return dictionary;

}

public static String spellChecker(String word, int threshold, String[] dictionary) {

int minerror = 15;

String min = "";

int temp = 15;

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

temp = minerror;

minerror = Math.min(levenshtein(dictionary[i], word), minerror);

if(minerror < temp ){

min = dictionary[i];

}

}

if(threshold >= minerror){

return min;

}else{

return word;

}

}

}