1. Ancient Cipher

Ancient Roman empire had a strong government system with various departments, including a secret service department. Important documents were sent between provinces and the capital in encrypted form to prevent eavesdropping. The most popular ciphers in those times were so called *substitution cipher* and *permutation cipher*.

Substitution cipher changes all occurrences of each letter to some other letter. Substitutes for all letters must be different. For some letters substitute letter may coincide with the original letter. For example, applying substitution cipher that changes all letters from A to Y to the next ones in the alphabet, and changes Z to A, to the message "VICTORIOUS" one gets the message "WJDUPSJPVT".

Permutation cipher applies some permutation to the letters of the message. For example, applying the permutation ⟨2, 1, 5, 4, 3, 7, 6, 10, 9, 8⟩ to the message "VICTORIOUS" one gets the message "IVOTCIRSUO".

It was quickly noticed that being applied separately, both substitution cipher and permutation cipher were rather weak. But when being combined, they were strong enough for those times. Thus, the most important messages were first encrypted using substitution cipher, and then the result was encrypted using permutation cipher. Encrypting the message "VICTORIOUS" with the combination of the ciphers described above one gets the message "JWPUDJSTVP".

Archeologists have recently found the message engraved on a stone plate. At the first glance it seemed completely meaningless, so it was suggested that the message was encrypted with some substitution and permutation ciphers. They have conjectured the possible text of the original message that was encrypted, and now they want to check their conjecture. They need a computer program to do it, so you have to write one.

Input:

Input contains several test cases. Each of them consists of two lines. The first line contains the message engraved on the plate. Before encrypting, all spaces and punctuation marks were removed, so the encrypted message contains only capital letters of the English alphabet. The second line contains the original message that is conjectured to be encrypted in the message on the first line. It also contains only capital letters of the English alphabet. The lengths of both lines of the input are equal and do not exceed 100.

Output:

For each test case, print one output line. Output YES if the message on the first line of the input could be the result of encrypting the message on the second line, or NO in the other case.

Sample Input:

```
JWPUDJSTVP
VICTORIOUS
MAMA
ROME
HAHA
HEHE
AAA
AAA
NEERCISTHEBEST
SECRETMESSAGES
```

After the last test case, there will be a new-line marks the end of input.

Sample Output:

```
YES
NO
YES
YES
NO
```

2. List of Conquests 2

In Act I, Leporello is telling Donna Elvira about his master's long list of conquests:

"This is the list of the beauties my master has loved, a list I've made out myself: take a look, read it with me. In Italy six hundred and forty, in Germany two hundred and thirty-one, a hundred in France, ninety-one in Turkey; but in Spain already a thousand and three! Among them are country girls, waiting-maids, city beauties; there are countesses, baronesses, marchionesses, princesses: women of every rank, of every size, of every age." (Madamina, il catalogo questo)

As Leporello records all the "beauties" Don Giovanni "loved" in chronological order, it is very troublesome for him to present his master's conquest to others because he needs to count the number of "beauties" by their nationality each time. You are to help Leporello to count.

Input:

The input consists of at most 2000 lines. The first line contains a number n, indicating that there will be n more lines. Each following line, with at most 75 characters, contains a country (the first word) and the name of a woman (the rest of the words in the line) Giovanni loved. You may assume that the names of all countries consist of only one word.

Output:

The output consists of lines that first sorted by total numbers greater to lower, then sort country names in alphabetical order if total numbers are equal. Each line starts with the name of a country, followed by the total number of women Giovanni loved in that country, separated by a space.

Sample Input:

```
3
Spain Donna Elvira
England Jane Doe
Spain Donna Anna
```

Sample Output:

```
Spain 2
England 1
```

Hint:

To read the input, one can use **`getchar()`** or string:

```
while (input != '\n')
{
    input = getchar();
}
```

3. Maze Traversal 2

For a given maze, there's a simple algorithm for walking through the maze that guarantees finding the exit (assuming there's an exit). If there's not an exit, you'll arrive at the starting location again. Place your **right** hand on the wall to your right and begin walking forward. Never remove your hand from the wall. If the maze turns to the right, you follow the wall to the right. As long as you do not remove your hand from the wall, eventually you'll arrive at the exit of the maze.

Input:

1. The input is a 15-by-15 maze.
2. The maze is guaranteed to have an entry at the left and an exit on the right (if there exists one exit).
3. The '#' symbols represent the walls where you cannot walk through.
4. The 'X' symbols represent the possible paths.

Sample Input 1:	Sample Input 2:
############### #XXXXXXXXX#XXX# #X#X#####X###X# #X#X#XXXXXXX#X# #X########X###X# #X#X#XXXXXXXXX# ###X###X####### #XXXXXXX#X#XXX# #X########X#X### XX#XXXXX#XXX#XX #X#X###X#X#X#X# #XXX#XXXXX#XXX# #X#####X######X# #XXXXX#XXXXX#X# ###############	############### #X#XXXXXXXXX#X# #X#X#X###X#X#X# #XXX#X#XXX#XXX# #########X###X# XXXXXXXXXXXXX#X# ###X########X### #XXXXXXXXX#XXX# #X#X#####X###X# #X#X#XXXXXXX#X# #X#########X###X# #X#X#XXXXXXXXX# ###X###X####### #XXX###XXXXXXX# ###############

Output:

1. Please follow the given algorithm to solve and show the solution of the maze.
2. Other solutions, for example, a shortest path is not acceptable.
3. Put 'R' symbols on the path that is taken and show the whole traversed maze as the result.
4. If the maze does have a solution, print "This maze has a solution" at the end.
5. If the maze does not have a solution, you still need to show the path and then print "This maze has no solution".

Sample Output 1:	Sample Output 2:
##############	##############
#XXXXXXXXX#XXX#	#R#RRRRRRRR#R#
#X#X#####X###X#	#R#R#R###R#R#R#
#X#X#XXXXXXX#X#	#RRR#R#RRR#RRR#
#X#######X###X#	#########R###R#
#X#X#XXXXXXXXX#	RRRRRRRRRRRR#R#
###X###X#######	###R#######R###
#XXXXXXX#X#XXX#	#RRRRRRRRR#RRR#
#X#######X#X###	#R#R#####R###R#
RR#RRRRR#RRR#RR	#R#R#RRRRRXX#R#
#R#R###R#R#R#R#	#R#######R###R#
#RRR#RRRRR#RRR#	#R#X#RRRRRRRRR#
#R#####R#####R#	###X###R#######
#RRRRR#RRRRR#R#	#XXX###RRRRRRR#
##############	##############
This maze has a solution	This maze has no solution

Hints:

1. To read the maze, one can use **`getchar()`** or string.
2. Using Two-dimensional Array can help to visualize the maze and the coordinate system.