1.) Robert  is having 2 strings consist of uppercase & lowercase english letters. Now he want to compare those two strings lexicographically. The letters' case does not matter, that is an uppercase letter is considered equivalent to the corresponding lowercase letter.

Input

The first line contains **T**. Then **T** test cases follow.

Each test case contains a two lines contains a string. The strings' lengths range from 1 to 100 inclusive. It is guaranteed that the strings are of the same length and also consist of uppercase and lowercase Latin letters.

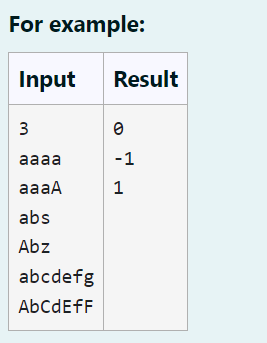
Output

If the first string is less than the second one, print "-1".  
If the second string is less than the first one, print "1".  
If the strings are equal, print "0".  
Note that the letters' case is not taken into consideration when the strings are compared.

Constraints

**1**≤**T**≤**50**

**String length**≤**100**



Program:

T = int(input().strip())

for \_ in range(T):

str1 = input().strip().lower()

str2 = input().strip().lower()

if str1 < str2:

print(-1)

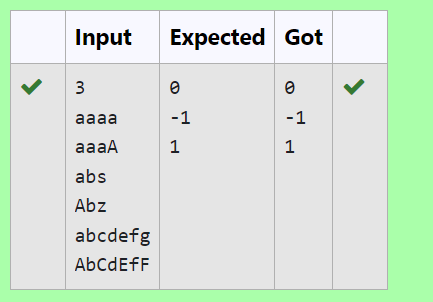
elif str1 > str2:

print(1)

else:

print(0)

OUTPUT:



2.) Given two Strings s1 and s2, remove all the characters from s1 which is present in s2.

**Constraints**

1<= string length <= 200

**Sample Input 1**

experience

enc

**Sample Output 1**

xpri

PROGRAM:

a=input()

b=input()

c=""

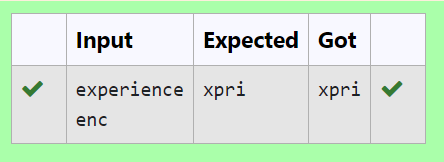
for char in a:

if char not in b:

c=c+char

print(c)

OUTPUT:



3.) Given a string S which is of the format USERNAME@DOMAIN.EXTENSION, the program must print the EXTENSION, DOMAIN, USERNAME in the reverse order.

**Input Format:**

The first line contains S.

**Output Format:**

The first line contains EXTENSION.  
The second line contains DOMAIN.  
The third line contains USERNAME.

**Boundary Condition:**

1 <= Length of S <= 100

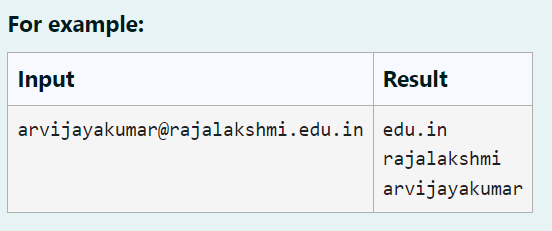
**Example Input/Output 1:**

Input:

abcd@gmail.com

Output:

com  
gmail  
abcd



PROGRAM:

s = input()

username\_domain, extension = s.split('@')

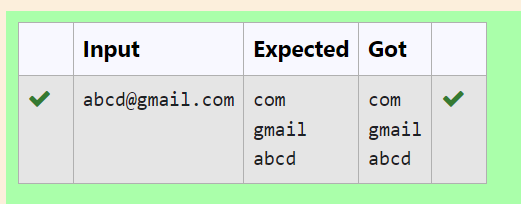
domain, extension = extension.split('.',1)

print(extension)

print(domain)

print(username\_domain)

OUTPUT:



4.) Consider the below words as key words and check the given input is key word or not.

keywords: {break, case, continue, default, defer, else, for, func, goto, if, map, range, return, struct, type, var}

Input format:

Take string as an input from stdin.

Output format:

Print the word is key word or not.

Example Input:

break

Output:

break is a keyword

Example Input:

IF

Output:

IF is not a keyword

**For example:**

| **Input** | **Result** |
| --- | --- |
| break | break is a keyword |
| IF | IF is not a keyword |

**PROGRAM:**

**a=input()**

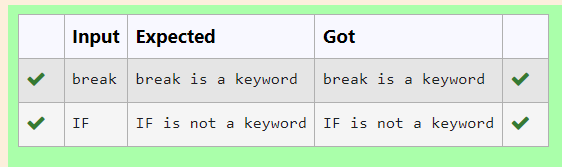
**if(a=="break" or a=="case" or a=="continue" or a=="default" or a=="defer" or a=="else" or a=="for" or a=="func" or a=="goto" or a=="map" or a=="if"or a=="range" or a=="return" or a=="struct" or a=="type" or a=="var"):**

**print(a,"is a keyword")**

**else:**

**print(a,"is not a keyword")**

**OUTPUT:**

****

5.) **.** Write a python program to count all letters, digits, and special symbols respectively from a given string

**For example:**

| **Input** | **Result** |
| --- | --- |
| rec@123 | 3  3  1 |

**PROGRAM:**

**a=input()**

**b=0**

**d=0**

**ss=0**

**for i in a:**

**if i.isalpha():**

**b+=1**

**elif i.isnumeric():**

**d+=1**

**else:**

**ss+=1**

**print(b)**

**print(d)**

**print(ss)**

**OUTPUT:**

****

6.) . **Reverse**a string **without affecting special characters**  
 Given a string **S**, containing special characters and all the alphabets, reverse the string without affecting the positions of the special characters.  
**Input:**A&B  
**Output:**B&A  
**Explanation**: As we ignore '&' and  
As we ignore '&' and then reverse, so answer is "B&A".

**For example:**

| **Input** | **Result** |
| --- | --- |
| A&x# | x&A# |

**PROGRAM:**

**s=input().strip()**

**ch=list(s)**

**left,right=0,len(ch)-1**

**while left<right:**

**if not ch[left].isalpha():**

**left+=1**

**elif not ch[right].isalpha():**

**right-=1**

**else:**

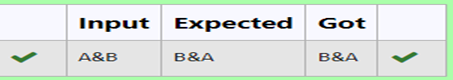
**ch[left],ch[right]=ch[right],ch[left]**

**left+=1**

**right-=1**

**print("".join(ch))**

**OUTPUT:**

****

7.) **.** Find if a String2 is substring of String1. If it is, return the index of the first occurrence. else return -1.

**Sample Input 1**

thistest123string

123

**Sample Output 1**

8

**PROGRAM:**

**a=input()**

**b=input()**

**# c=0**

**# for i in a:**

**#     if(i.isalpha()):**

**#         c+=1**

**#     else:**

**#         break**

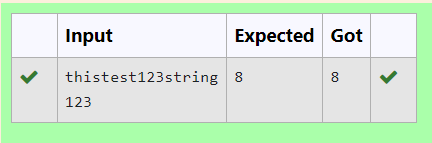
**if b in a:**

**print(a.index(b))**

**else:**

**print(-1)**

**OUTPUT:**

****

8.) Write a program that takes as input a string (sentence), and returns its second word in uppercase.

For example:

If input is “Wipro Technologies Bangalore” the function should return “TECHNOLOGIES”

If input is “Hello World” the function should return “WORLD”

If input is “Hello” the program should return “LESS”

NOTE 1: If input is a sentence with less than 2 words, the program should return the word “LESS”.

NOTE 2: The result should have no leading or trailing spaces.

**For example:**

| **Input** | **Result** |
| --- | --- |
| Wipro Technologies Bangalore | TECHNOLOGIES |
| Hello World | WORLD |
| Hello | LESS |

**PROGRAM:**

**def second\_word\_uppercase(sentence):**

**words = sentence.split()**

**if len(words) < 2:**

**return "LESS"**

**else:**

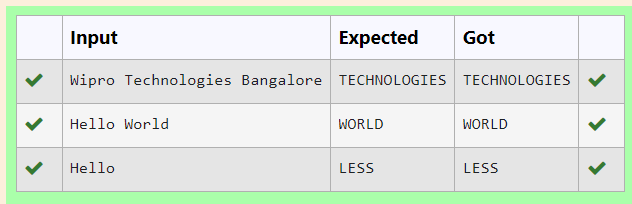
**return words[1].upper()**

**input\_sentence =str(input())**

**result= second\_word\_uppercase(input\_sentence)**

**print(result)**

**OUTPUT:**

****

9.) Write a program to check if two strings are balanced. For example, strings s1 and s2 are balanced if all the characters in the s1 are present in s2. The character’s position doesn’t matter. If balanced display as "true" ,otherwise "false".

**For example:**

| **Input** | **Result** |
| --- | --- |
| Yn  PYnative | True |

**PROGRAM:**

**def are\_strings\_balanced(s1, s2):**

**# Convert both strings to sets to remove duplicates**

**s1\_set = set(s1)**

**s2\_set = set(s2)**

**# Check if all characters in s1 are present in s2**

**return s1\_set.issubset(s2\_set)**

**# Example usage:**

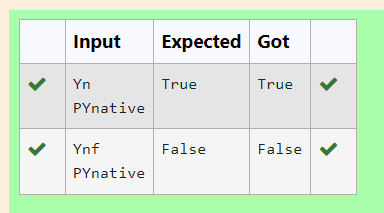
**s1 = input()**

**s2 = input()**

**result = are\_strings\_balanced(s1, s2)**

**print(result)  # Output: True**

**OUTPUT:**

****

10.) In this exercise, you will create a program that reads words from the user until the user enters a blank line. After the user enters a blank line your program should display each word entered by the user exactly once. The words should be displayed in the same order that they were first entered. For example, if the user enters:

first

second

first

third

second

then your program should display:

first

second

third

PROGRAM:

ww=[]

while True:

w=input()

if w==" ":

break

if w not in ww:

ww.append(w)

for i in ww:

print(i)

**OUTPUT:**

