Question 1

Bigger Is Greater

Lexicographical order is often known as alphabetical order when dealing with strings. A string is *greater* than another string if it comes later in a lexicographically sorted list.

Given a word , create a new word by swapping some or all of its characters. This new word must meet two criteria:

- It must be greater than the original word
- It must be the smallest word that meets the first condition

For example, given the word **w = abcd** , the next largest word is **abdc.**

Complete the function *BiggerIsGreater* below to create and return the new string meeting the criteria. If it is not possible, return no answer.

Function Description

Complete the *BiggerIsGreater* function in the editor below. It should return the smallest lexicographically higher string possible from the given string or no answer.

BiggerIsGreater has the following parameter(s):

- *w*: a string

Input Format

The first line of input contains ***T*** , the number of test cases.

Each of the next ***T*** lines contains ***w*** .

Constraints

- $1 \leq T \leq 10^5$
- $1 \leq |w| \leq 100$
- ***w*** will contain only letters in the range ascii[a..z].

Output Format

For each test case, output the string meeting the criteria. If no answer exists, print no answer.

Sample Input 0

```
5
ab
bb
hefg
dhck
dkhc
```

Sample Output 0

```
ba
no answer
hegf
dhkc
hcdk
```

Explanation 0

- *Test case 1:*
  ba is the only string which can be made by rearranging ab. It is greater.
- *Test case 2:*
  It is not possible to rearrange bb and get a greater string.
- *Test case 3:*
  hegf is the next string greater than hefg.
- *Test case 4:*
  dhkc is the next string greater than dhck.
- *Test case 5:*
  hcdk is the next string greater than dkhc.

Sample Input 1

```
5
lmno
dcba
dcbb
abdc
abcd
```

Sample Output 1

```
lmon
no answer
no answer
acbd
abdc
```

Question 2

Encryption

An English text needs to be encrypted using the following encryption scheme.
First, the spaces are removed from the text. Let L be the length of this text.
Then, characters are written into a grid, whose rows and columns have the following constraints:
$\lfloor\sqrt{L}\rfloor \leq$ row $\leq$ column $\leq \lceil\sqrt{L}\rceil$, where $\lfloor x \rfloor$ is a floor function

and $\lceil x \rceil$ is a ceil function

For example, the sentence s = if man was meant to stay on the ground god would have given us roots , after removing spaces is 54 characters long. $\sqrt{54}$ is between 7 and 8, so it is written in the form of a grid with 7 rows and 8 columns.

ifmanwas
meanttos
tayonthe
groundgo
dwouldha
vegivenu
sroots

8 → C.
7 R

aarushsinha

- Ensure that *rows x columns* $\geq L$
- If multiple grids satisfy the above conditions, choose the one with the minimum area, i.e *rows x columns* .

The encoded message is obtained by displaying the characters in a column, inserting a space, and then displaying the next column and inserting a space, and so on. For example, the encoded message for the above rectangle is:

imtgdvs fearwer mayoogo anouuio ntnnlvt wttddes aohghn sseoau

You will be given a message to encode and print.

Function Description

Complete the *encryption* function in the editor below. It should return a single string composed as described.

encryption has the following parameter(s):

- *s*: a string to encrypt

Input Format

One line of text, the string *s*.

Constraints

- $1 \leq |s| \leq 81$
- *s* is comprised only of characters in the range ascii[a-z].

Output Format

Print the encoded message on one line as described.

Sample Input 0

haveaniceday

Sample Output 0
hae and via ecy

Sample Input 1
feedthedog

Sample Output 1
fto ehg ee dd

Sample Input 2
chillout

Sample Output 2
clu hlt io

Question 3

Postfix And Infix

Stack is a data structure that has a main application of evaluating arithmetic expressions in systems.

The data input is in *Infix* form and gets converted it in *Postfix* form for example the expression given,

Infix = (A+B)*(C+D) , will first evaluate A+B and separately evaluate C+D

and treat them as variables, AB+ and CD+ ,

final expression will be treated for AB+ and CD+ as variables will be evaluated as AB+CD+*.

This is done using *push* and *pop* operations of a stack. Adding elements in the stack is the *push*

operation and deleting elements is *pop* operations

Given an input in *Infix* form convert it into *Postfix* form , create the function *PostfixInfix* to convert

an infix input and display in postfix form.

Function Description

Create the *PostfixInfix* function which should return

the postfix form of the infix input .

PostfixInfix has parameter(s)

- *i* : an expression
- *b* : operands of expression

Input Format

The first line contains *i* , the infix expression,

with *b* operands.

Constraints

- $2 \leq b \leq 6$
- b will contain only letters in the range ascii[a..g].

Output Format

For each test case , output string meeting the criteria.

Sample input 0

(a-b)*(c/d)+e

Sample Output 0

ab-cd/e*

Explanation 0

(	Push '('	#(	
a	Display 'a'	#(	a
-	Push '-'	#(-	a
b	Display 'b'	#(-	ab
)	Pop and Display	#(	ab-
		#empty	
*	Push '*'	#*	ab-
(	Push '('	#*(	ab-
c	Display 'c'	#*(	ab-c
/	Push '/'	#*(/	ab-c
d	Display 'd'	#*(/	ab-cd
)	Pop and Display '/'	#*(	ab-cd/
	Pop	#*	
+	Pop and Display '*'	#	ab-cd/*
	Push '+'	#+	ab-cd/*
e	Display 'e'	#+	ab-cd/*e
	Pop and Dislpay	#	ab-cd/*e+
	end		

Sample Input 1

(a*(b+(c+d)*(e+f)/g))*h

Sample Output 1

abcd+ef+*g/+*h*

Question-4

Total Question Marks

Create a function which will accept strings, and will contain single digit numbers, letters, and question marks, and check if there are exactly 3 question marks between every pair of two numbers that add up to 10. If so, then your program should return the string true, otherwise it should return the string false. If there aren't any two numbers that add up to 10 in the string,

then your program should return false as well.

For example: if the string is *s = "arrb6???4xxb15???eee5 "* then your program should return *True* because there are exactly 3 question marks between 6 and 4, and 3 question marks between 5 and 5 at the end of the string.

Input Format

The first line contains *T*, the number of test cases.

Each of the next *T* lines contains a string *s* , which contains different characters.

Constraints

s will contain only letters in the range asci[a..z]

Output Format

For each test case, output False or True with respect to each string.

Sample Input 0

aa6?9

Sample Output 0

False

Sample Input 1

acc?7??sss?3rr1????5

Sample Output 1

True

Ans: