Codeforces Round #739 (Div. 3)

A. Dislike of Threes

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp doesn't like integers that are divisible by 3 or end with the digit 3 in their decimal representation. Integers that meet both conditions are disliked by Polycarp, too.

Polycarp starts to write out the positive (greater than 0) integers which he likes: $1, 2, 4, 5, 7, 8, 10, 11, 14, 16, \ldots$. Output the k-th element of this sequence (the elements are numbered from 1).

Input

The first line contains one integer t ($1 \le t \le 100$) — the number of test cases. Then t test cases follow.

Each test case consists of one line containing one integer k ($1 \le k \le 1000$).

Output

For each test case, output in a separate line one integer x — the k-th element of the sequence that was written out by Polycarp.

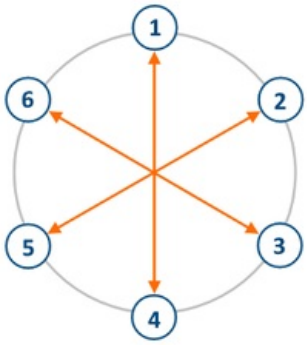
Example

input
10 1 2 3 4 5 6 7 8 9 1000
output
1 2 4 5 7 8 10 11 14 1666

B. Who's Opposite?

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Some number of people (this number is even) have stood in a circle. The people stand in the circle evenly. They are numbered clockwise starting from a person with the number 1. Each person is looking through the circle's center at the opposite person.



A sample of a circle of 6 persons. The orange arrows indicate who is looking at whom.

You don't know the exact number of people standing in the circle (but this number is even, no doubt). It is known that the person

with the number a is looking at the person with the number b (and vice versa, of course). What is the number associated with a person being looked at by the person with the number c? If, for the specified a, b, and c, no such circle exists, output -1.

Input

The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases. Then t test cases follow.

Each test case consists of one line containing three **distinct** integers a, b, c ($1 \le a, b, c \le 10^8$).

Output

For each test case output in a separate line a single integer d — the number of the person being looked at by the person with the number c in a circle such that the person with the number a is looking at the person with the number b. If there are multiple solutions, print any of them. Output -1 if there's no circle meeting the given conditions.

Example

input
7 6 2 4 2 3 1 2 4 10 5 3 4 1 3 2 2 5 4 4 3 2
output
8 -1 -1 -1 4 1 -1

Note

In the first test case, there's a desired circle of 8 people. The person with the number 6 will look at the person with the number 2 and the person with the number 8 will look at the person with the number 4.

In the second test case, there's no circle meeting the conditions. If the person with the number 2 is looking at the person with the number 3, the circle consists of 2 people because these persons are neighbors. But, in this case, they must have the numbers 1 and 2, but it doesn't meet the problem's conditions.

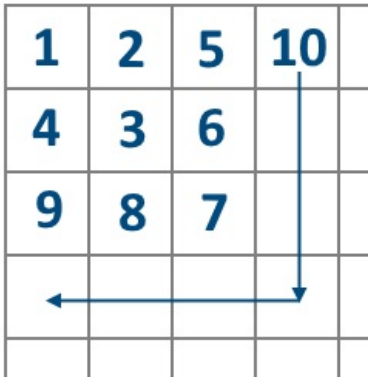
In the third test case, the only circle with the persons with the numbers 2 and 4 looking at each other consists of 4 people. Therefore, the person with the number 10 doesn't occur in the circle.

C. Infinity Table

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp has found a table having an infinite number of rows and columns. The rows are numbered from 1, starting from the topmost one. The columns are numbered from 1, starting from the leftmost one.

Initially, the table hasn't been filled and Polycarp wants to fix it. He writes integers from 1 and so on to the table as follows.



The figure shows the placement of the numbers from 1 to 10. The following actions are denoted by the arrows.

The leftmost topmost cell of the table is filled with the number 1. Then he writes in the table all positive integers beginning from 2 sequentially using the following algorithm.

First, Polycarp selects the leftmost non-filled cell in the first row and fills it. Then, while the left neighbor of the last filled cell is filled, he goes down and fills the next cell. So he goes down until the last filled cell has a non-filled neighbor to the left (look at the vertical arrow going down in the figure above).

After that, he fills the cells from the right to the left until he stops at the first column (look at the horizontal row in the figure above).

Then Polycarp selects the leftmost non-filled cell in the first row, goes down, and so on.

A friend of Polycarp has a favorite number k. He wants to know which cell will contain the number. Help him to find the indices of the row and the column, such that the intersection of the row and the column is the cell containing the number k.

Input
The first line contains one integer t ($1 \le t \le 100$) — the number of test cases. Then t test cases follow.

Each test case consists of one line containing one integer k ($1 \le k \le 10^9$) which location must be found.

Output
For each test case, output in a separate line two integers r and c ($r, c \ge 1$) separated by spaces — the indices of the row and the column containing the cell filled by the number k, respectively.

Example

input
7 11 14 5 4 1 2 1000000000
output
2 4 4 3 1 3 2 1 1 1 1 2 31623 14130

D. Make a Power of Two

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an integer n. In 1 move, you can do one of the following actions:

- erase any digit of the number (it's acceptable that the number before the operation has exactly one digit and after the operation, it is "empty");
- add one digit **to the right**.

The actions may be performed in any order any number of times.

Note that if, after deleting some digit from a number, it will contain leading zeroes, they will **not** be deleted. E.g. if you delete from the number 301 the digit 3, the result is the number 01 (not 1).

You need to perform the **minimum** number of actions to make the number any power of 2 (i.e. there's an integer k ($k \ge 0$) such that the resulting number is equal to 2^k). **The resulting number must not have leading zeroes**.

E.g. consider $n = 1052$. The answer is equal to 2. First, let's add to the right one digit 4 (the result will be 10524). Then let's erase the digit 5, so the result will be 1024 which is a power of 2.

E.g. consider $n = 8888$. The answer is equal to 3. Let's erase any of the digits 8 three times. The result will be 8 which is a power of 2.

Input
The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases. Then t test cases follow.

Each test case consists of one line containing one integer n ($1 \le n \le 10^9$).

Output
For each test case, output in a separate line one integer m — the minimum number of moves to transform the number into any power of 2.

Example

input
12 1052 8888 6 75 128 1 301

12048
1504
6656
1000000000
687194767

output

2
3
1
3
0
0
2
1
3
4
9
2

Note

The answer for the first test case was considered above.

The answer for the second test case was considered above.

In the third test case, it's enough to add to the right the digit 4 — the number 6 will turn into 64.

In the fourth test case, let's add to the right the digit 8 and then erase 7 and 5 — the taken number will turn into 8.

The numbers of the fifth and the sixth test cases are already powers of two so there's no need to make any move.

In the seventh test case, you can delete first of all the digit 3 (the result is 01) and then the digit 0 (the result is 1).

E. Polycarp and String Transformation

time limit per test: 3.0 s
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp has a string s. Polycarp performs the following actions until the string s is empty (t is initially an empty string):

- he adds to the right to the string t the string s, i.e. he does $t = t + s$, where $t + s$ is a concatenation of the strings t and s;
- he selects an arbitrary letter of s and removes from s all its occurrences (**the selected letter must occur in the string s at the moment of performing this action**).

Polycarp performs this sequence of actions **strictly** in this order.

Note that after Polycarp finishes the actions, the string s will be empty and the string t will be equal to some value (that is undefined and depends on the order of removing).

E.g. consider s="abacaba" so the actions may be performed as follows:

- t="abacaba", the letter 'b' is selected, then s="aacaa";
- t="abacabaaacaa", the letter 'a' is selected, then s="c";
- t="abacabaaacaac", the letter 'c' is selected, then s="" (the empty string).

You need to restore the initial value of the string s using only the final value of t and find the order of removing letters from s.

Input

The first line contains one integer T ($1 \le T \le 10^4$) — the number of test cases. Then T test cases follow.

Each test case contains one string t consisting of lowercase letters of the Latin alphabet. The length of t doesn't exceed $5 \cdot 10^5$. The sum of lengths of all strings t in the test cases doesn't exceed $5 \cdot 10^5$.

Output

For each test case output in a separate line:

- -1, if the answer doesn't exist;
- two strings separated by spaces. The first one must contain a possible initial value of s. The second one must contain a sequence of letters — it's in what order one needs to remove letters from s to make the string t. E.g. if the string "bac" is outputted, then, first, all occurrences of the letter 'b' were deleted, then all occurrences of 'a', and then, finally, all occurrences of 'c'. If there are multiple solutions, print any one.

Example

input

7
abacabaaacaac
nowyouknowthat
polycarppoycarppoyarppyarppyrpprppp

isi everywherevrywhrvryhrvrhrvhv haaha qweqeewew

output
abacaba bac -1 polycarp lcoayrp is si everywhere ewyrhv -1 -1

Note
The first test case is considered in the statement.

F1. Nearest Beautiful Number (easy version)

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

It is a simplified version of problem F2. The difference between them is the constraints (F1: $k \le 2$, F2: $k \le 10$).

You are given an integer n. Find the minimum integer x such that $x \ge n$ and the number x is k-*beautiful*.

A number is called k-*beautiful* if its decimal representation having no leading zeroes contains no more than k different digits. E.g. if $k = 2$, the numbers 3434443, 55550, 777 and 21 are k-*beautiful* whereas the numbers 120, 445435 and 998244353 are not.

Input
The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases. Then t test cases follow.

Each test case consists of one line containing two integers n and k ($1 \le n \le 10^9$, $1 \le k \le 2$).

Output
For each test case output on a separate line x — the minimum k-*beautiful* integer such that $x \ge n$.

Example

input
4 1 1 221 2 177890 2 998244353 1

output
1 221 181111 999999999

F2. Nearest Beautiful Number (hard version)

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

It is a complicated version of problem F1. The difference between them is the constraints (F1: $k \le 2$, F2: $k \le 10$).

You are given an integer n. Find the minimum integer x such that $x \ge n$ and the number x is k-*beautiful*.

A number is called k-*beautiful* if its decimal representation having no leading zeroes contains no more than k different digits. E.g. if $k = 2$, the numbers 3434443, 55550, 777 and 21 are k-*beautiful* whereas the numbers 120, 445435 and 998244353 are not.

Input
The first line contains one integer t ($1 \le t \le 10^4$) — the number of test cases. Then t test cases follow.

Each test case consists of one line containing two integers n and k ($1 \le n \le 10^9$, $1 \le k \le 10$).

Output
For each test case output on a separate line x — the minimum k-*beautiful* integer such that $x \ge n$.

Example

input
6 2021 3

177890 2
34512 3
724533 4
998244353 1
12345678 10

output

2021
181111
34533
724542
999999999
12345678