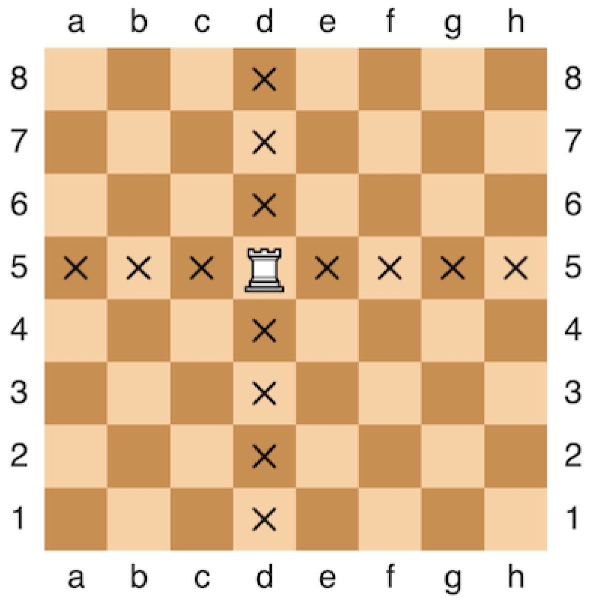
# Codeforces Round 913 (Div. 3)

## A. Rook

1 second, 256 megabytes

As you probably know, chess is a game that is played on a board with 64 squares arranged in an $8 \times 8$ grid. Columns of this board are labeled with letters from **a** to **h**, and rows are labeled with digits from **1** to **8**. Each square is described by the row and column it belongs to.



The rook is a piece in the game of chess. During its turn, it may move any non-zero number of squares horizontally or vertically. Your task is to find all possible moves for a rook on an empty chessboard.

### Input

The first line of input contains single integer $t$ $(1 \le t \le 64)$ — the number of test cases. The descriptions of test cases follow.

Each test case contains one string of two characters, description of the square where rook is positioned. The first character is a letter from **a** to **h**, the label of column, and the second character is a digit from **1** to **8**, the label of row.

The same position may occur in more than one test case.

### Output

For each test case, output descriptions of all squares where the rook can move, in the same format as in the input.

You can output squares in any order per test case.

input
```
1
d5
```

output
```
d1
d2
b5
g5
h5
d3
e5
f5
d8
a5
d6
d7
c5
d4
```

## B. YetnotherrokenKeoard

1 second, 256 megabytes

Polycarp has a problem — his laptop keyboard is broken.

Now, when he presses the 'b' key, it acts like an unusual backspace: it deletes the last (rightmost) lowercase letter in the typed string. If there are no lowercase letters in the typed string, then the press is completely ignored.

Similarly, when he presses the 'B' key, it deletes the last (rightmost) uppercase letter in the typed string. If there are no uppercase letters in the typed string, then the press is completely ignored.

In both cases, the letters 'b' and 'B' are not added to the typed string when these keys are pressed.

Consider an example where the sequence of key presses was "ARaBbbitBaby". In this case, the typed string will change as follows: "" $\xrightarrow{\texttt{A}}$ "A" $\xrightarrow{\texttt{R}}$ "AR" $\xrightarrow{\texttt{a}}$ "ARa" $\xrightarrow{\texttt{B}}$ "Aa" $\xrightarrow{\texttt{b}}$ "A" $\xrightarrow{\texttt{b}}$ "A" $\xrightarrow{\texttt{i}}$ "Ai" $\xrightarrow{\texttt{t}}$ "Ait" $\xrightarrow{\texttt{B}}$ "it" $\xrightarrow{\texttt{a}}$ "ita" $\xrightarrow{\texttt{b}}$ "it" $\xrightarrow{\texttt{y}}$ "ity".

Given a sequence of pressed keys, output the typed string after processing all key presses.

### Input

The first line of the input data contains an integer $t$ $(1 \le t \le 1000)$, the number of test cases in the test.

The following contains $t$ non-empty lines, which consist of lowercase and uppercase letters of the Latin alphabet.

It is guaranteed that each line contains at least one letter and the sum of the lengths of the lines does not exceed $10^6$.

### Output

For each test case, output the result of processing the key presses on a separate line. If the typed string is empty, then output an empty line.

input
```
12
ARaBbbitBaby
YetAnotherBrokenKeyboard
Bubble
Improbable
abbreviable
BbBB
BusyasaBeeinaBedofBloomingBlossoms
CoDEBARbIES
codeforces
bobebobbes
b
TheBBlackbboard
```

output
```
ity
YetnotherrokenKeoard
le
Imprle
revile

usyasaeeinaedofloominglossoms
CDARIES
codeforces
es

helaoard
```

## C. Removal of Unattractive Pairs

1 second, 256 megabytes

Vlad found a string $s$ consisting of $n$ lowercase Latin letters, and he wants to make it as short as possible.

To do this, he can remove **any** pair of adjacent characters from $s$ any number of times, provided they are **different**. For example, if $s$=`racoon`, then by removing one pair of characters he can obtain the strings `coon`, `roon`, `raon`, and `raco`, but he cannot obtain `racn` (because the removed letters were the same) or `rcon` (because the removed letters were not adjacent).

What is the minimum length Vlad can achieve by applying any number of deletions?

### Input

The first line of the input contains a single integer $t$ ($1 \le t \le 10^4$) — the number of test cases. Descriptions of the test cases follow.

The first line of each test case contains a single integer $n$ ($1 \le n \le 2 \cdot 10^5$) — the length of the string $s$.

The second line of each test case contains the string $s$ consisting of $n$ lowercase Latin letters.

It is guaranteed that the sum of $n$ over all test cases does not exceed $2 \cdot 10^5$.

### Output

For each test case, output a single number—the minimum length of the string $s$, after removing pairs of adjacent characters with different values.

input

```
10
4
aabc
5
abaca
10
avbvvcvvvd
7
abcdefg
5
dabbb
8
aacebeaa
7
bbbbacc
6
dacfcc
6
fdfcdc
9
dbdcfbbdc
```

output

```
0
1
2
1
1
0
1
0
0
1
```

In the first test case of the example, you need to act as follows: "`aabc`" → "`ac`" → "". Note that with a different order of deletions, the string will not become empty.

## D. Jumping Through Segments

5 seconds, 256 megabytes

Polycarp is designing a level for a game. The level consists of $n$ segments on the number line, where the $i$-th segment starts at the point with coordinate $l_i$ and ends at the point with coordinate $r_i$.

The player starts the level at the point with coordinate $0$. In one move, they can move to any point that is within a distance of no more than $k$. After their $i$-th move, the player must land within the $i$-th segment, that is, at a coordinate $x$ such that $l_i \le x \le r_i$. This means:

- After the first move, they must be inside the first segment (from $l_1$ to $r_1$);
- After the second move, they must be inside the second segment (from $l_2$ to $r_2$);
- ...
- After the $n$-th move, they must be inside the $n$-th segment (from $l_n$ to $r_n$).

The level is considered completed if the player reaches the $n$-th segment, following the rules described above. After some thought, Polycarp realized that it is impossible to complete the level with some values of $k$.

Polycarp does not want the level to be too easy, so he asks you to determine the minimum integer $k$ with which it is possible to complete the level.

### Input

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

The first line of each test case contains a single integer $n$ ($1 \le n \le 2 \cdot 10^5$)—the number of segments in the level.

The following $n$ lines.

The $i$-th line contain two integers $l_i$ and $r_i$ ($0 \le l_i \le r_i \le 10^9$)—the boundaries of the $i$-th segment. Segments may intersect.

It is guaranteed that the sum of $n$ over all test cases does not exceed $2 \cdot 10^5$.

### Output

For each test case, output a single integer—the minimum value of $k$ with which it is possible to complete the level.

input

```
4
5
1 5
3 4
5 6
8 10
0 1
3
0 2
0 1
0 3
3
3 8
10 18
6 11
4
10 20
0 5
15 17
2 2
```

output

```
7
0
5
13
```

## E. Good Triples

2 seconds, 256 megabytes

Given a non-negative integer number $n$ ($n \ge 0$). Let's say a triple of non-negative integers $(a, b, c)$ is good if $a + b + c = n$, and $digsum(a) + digsum(b) + digsum(c) = digsum(n)$, where $digsum(x)$ is the sum of digits of number $x$.

For example, if $n = 26$, then the pair $(4, 12, 10)$ is good, because $4 + 12 + 10 = 26$, and $(4) + (1 + 2) + (1 + 0) = (2 + 6)$.

Your task is to find the number of good triples for the given number $n$. The order of the numbers in a triple matters. For example, the triples $(4, 12, 10)$ and $(10, 12, 4)$ are two different triples.

### Input

The first line of input contains a single integer $t$ ($1 \le t \le 10^4$) — the number of test cases. Descriptions of test cases follow.

The first and only line of the test case contains one integer $n$ ($0 \le n \le 10^7$).

**Output**

For each test case output one integer, the number of good triples for the given number $n$. Order of numbers in a triple

```
input
12
11
0
1
2
3
4
5
3141
999
2718
9999999
10000000
```

```
output
9
1
3
6
10
15
21
1350
166375
29160
1522435234375
3
```

In the first example, the good triples are $(0, 0, 11)$, $(0, 1, 10)$, $(0, 10, 1)$, $(0, 11, 0)$, $(1, 0, 10)$, $(1, 10, 0)$, $(10, 0, 1)$, $(10, 1, 0)$, $(11, 0, 0)$.

In the second example, there is only one good triple $(0, 0, 0)$.

## F. Shift and Reverse

2.0 s, 256 megabytes

Given an array of integers $a_1, a_2, \ldots, a_n$. You can make two types of operations with this array:

- Shift: move the last element of array to the first place, and shift all other elements to the right, so you get the array $a_n, a_1, a_2, \ldots, a_{n-1}$.
- Reverse: reverse the whole array, so you get the array $a_n, a_{n-1}, \ldots, a_1$.

Your task is to sort the array in non-decreasing order using the minimal number of operations, or say that it is impossible.

**Input**

The first line of input contains a single integer $t$ ($1 \le t \le 10^4$) — the number of test cases. Descriptions of test cases follow.

The first line of each test case contains an integer $n$ ($1 \le n \le 10^5$) — size of the array.

The second line of each test case contains $n$ integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — elements of the array.

It is guaranteed that sum of $n$ over all test cases does not exceed $2 \cdot 10^5$.

**Output**

For each test case output the number $k$, the minimal number of operations you need to sort the array. If it is impossible to sort the array using these operations, output $-1$.

```
input
11
5
3 2 1 5 4
5
1 1 2 1 1
4
3 7 10 5
5
1 2 3 4 5
2
5 1
3
3 4 1
5
4 1 3 4 4
3
5 1 1
4
2 5 5 4
5
2 2 1 1 2
2
5 5
```

```
output
3
2
-1
0
1
1
3
1
2
2
0
```

In the first test case of the example, to sort the array $[3, 2, 1, 5, 4]$ you need to perform $3$ operations:

- Shift to obtain the array $[4, 3, 2, 1, 5]$;
- Shift to obtain the array $[5, 4, 3, 2, 1]$;
- Reverse to obtain the array $[1, 2, 3, 4, 5]$.

In the third test case of the example, it can be shown that it is impossible to sort the array using the given operations.

In the sixth test case of the example, to sort the array $[4, 1, 3, 4, 4]$ you need to perform $3$ operations:

- Reverse to obtain the array $[4, 4, 3, 1, 4]$;
- Shift to obtain the array $[4, 4, 4, 3, 1]$;
- Reverse to obtain the array $[1, 3, 4, 4, 4]$.

## G. Lights

2 seconds, 256 megabytes

In the end of the day, Anna needs to turn off the lights in the office. There are $n$ lights and $n$ light switches, but their operation scheme is really strange. The switch $i$ changes the state of light $i$, but it also changes the state of some other light $a_i$ (change the state means that if the light was on, it goes off and vice versa).

Help Anna to turn all the lights off using minimal number of switches, or say it is impossible.

**Input**

The first line of input contains a single integer $t$ ($1 \le t \le 10^4$) — the number of test cases. Descriptions of test cases follow.

The first line of each test case contains the integer $n$ ($2 \le n \le 10^5$) — the number of lights.

The second line of each test case contains the string of $n$ characters, the initial state of the lights. Character "`0`" means that the corresponding light is off, and "`1`" means that it is on.

The third line of each test case contains $n$ integers $a_i$ ($1 \le a_i \le n$, $a_i \ne i$) — the switch $i$ changes the states of light $i$ and light $a_i$.

It is guaranteed that sum of $n$ over all test cases does not exceed $2 \cdot 10^5$

### Output

For each test case output the integer $k$, the minimal number of switches to use, then in the separate line output the list of $k$ switches.

If it is impossible to turn off all the lights, output single integer $-1$.

input

```
8
5
11101
4 3 4 2 2
2
10
2 1
10
0000000011
9 10 10 7 10 9 9 9 10 2
10
1000111101
9 3 8 9 2 1 3 7 2 7
10
0001101010
5 7 6 10 8 3 6 6 2 2
10
0101100010
8 7 7 9 9 4 1 4 2 7
10
1010111010
7 9 10 7 7 2 8 6 10 4
10
1110000001
3 10 10 1 10 8 6 3 2 1
```

output

```
3
1 5 3
-1
1
9
5
5 6 10 2 3
6
4 9 5 10 8 7
3
5 4 9
6
1 3 5 9 7 8
2
2 1
```