A. Stair, Peak, or Neither?

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

You are given three digits a, b, and c. Determine whether they form a stair, a peak, or neither.

- A *stair* satisfies the condition $a < b < c$.
- A *peak* satisfies the condition $a < b > c$.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases.

The only line of each test case contains three digits a, b, c ($0 \le a, b, c \le 9$).

Output

For each test case, output "STAIR" if the digits form a stair, "PEAK" if the digits form a peak, and "NONE" otherwise (output the strings without quotes).

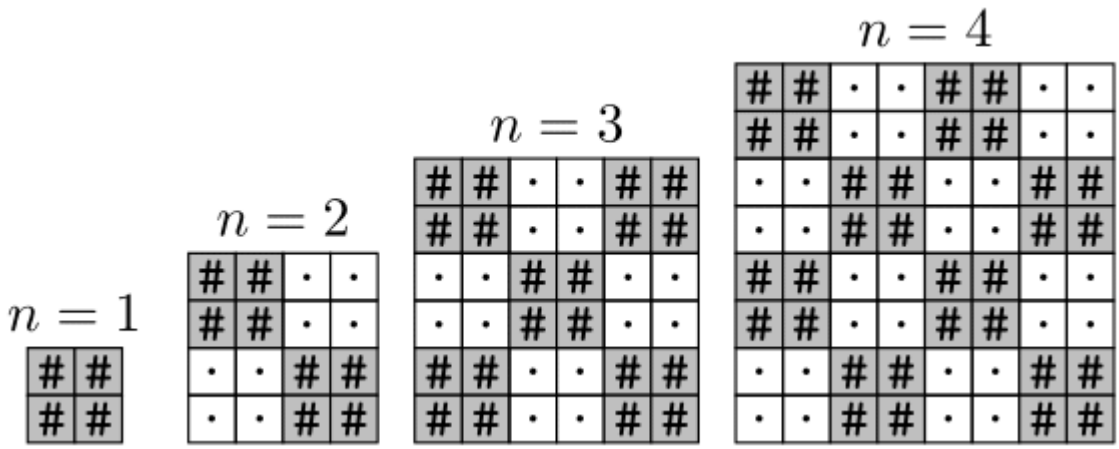
Standard Input	Standard Output
7 1 2 3 3 2 1 1 5 3 3 4 1 0 0 0 4 1 7 4 5 7	STAIR NONE PEAK PEAK NONE NONE STAIR

,

B. Upscaling

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 megabytes

You are given an integer n. Output a $2n \times 2n$ checkerboard made of 2×2 squares alternating '#' and '.', with the top-left cell being '#'.



The picture above shows the answers for $n = 1, 2, 3, 4$.

Input

The first line contains an integer t ($1 \le t \le 20$) — the number of test cases.

The only line of each test case contains a single integer n ($1 \le n \le 20$) — it means you need to output a checkerboard of side length $2n$.

Output

For each test case, output $2n$ lines, each containing $2n$ characters without spaces — the checkerboard, as described in the statement. Do **not** output empty lines between test cases.

Standard Input	Standard Output
4 1 2 3 4	## ## ##.. ##.. ..## ..## ##..## ##..## ..##.. ..##.. ##..## ##..## ##..##.. ##..##.. ..##..## ..##..##

	##..##.. ##..##.. ..##..## ..##..##

,

C. Clock Conversion

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

Given the time in 24-hour format, output the equivalent time in 12-hour format.

- 24-hour format divides the day into 24 hours from 00 to 23, each of which has 60 minutes from 00 to 59.
- 12-hour format divides the day into two halves: the first half is AM, and the second half is PM. In each half, the hours are numbered in the order $12, 01, 02, 03, \ldots, 11$. Each hour has 60 minutes numbered from 00 to 59.

Input

The first line contains a single integer t ($1 \le t \le 1440$) — the number of test cases.

The only line of each test case contains a string s of length 5 with format `hh:mm` representing a valid time in the 24-hour format. `hh` represents the hour from 00 to 23, and `mm` represents the minute from 00 to 59.

The input will always be a valid time in 24-hour format.

Output

For each test case, output two strings separated by a space ("`hh:mm AM`" or "`hh:mm PM`"), which are the 12-hour equivalent to the time provided in the test case (without quotes).

You should output the time exactly as indicated; in particular, you should not remove leading zeroes.

Standard Input	Standard Output
11 09:41 18:06 12:14 00:59 00:00 14:34 01:01 19:07 11:59 12:00 21:37	09:41 AM 06:06 PM 12:14 PM 12:59 AM 12:00 AM 02:34 PM 01:01 AM 07:07 PM 11:59 AM 12:00 PM 09:37 PM

,

D. Product of Binary Decimals

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 256 megabytes

Let's call a number a *binary decimal* if it is a positive integer and all digits in its decimal notation are either 0 or 1. For example, $1\,010\,111$ is a binary decimal, while $10\,201$ and $787\,788$ are not.

Given a number n, you are asked whether or not it is possible to represent n as a product of some (not necessarily distinct) binary decimals.

Input

The first line contains a single integer t ($1 \le t \le 5 \cdot 10^4$) — the number of test cases.

The only line of each test case contains a single integer n ($1 \le n \le 10^5$).

Output

For each test case, output "YES" (without quotes) if n can be represented as a product of binary decimals, and "NO" (without quotes) otherwise.

You can output "YES" and "NO" in any case (for example, strings "yES", "yes", and "Yes" will be recognized as a positive response).

Standard Input	Standard Output
11 121 1 14641 12221 10110 100000 99 112 2024 12421 1001	YES YES YES YES YES YES NO NO NO NO YES

Note

The first five test cases can be represented as a product of binary decimals as follows:

- $121 = 11 \times 11$.
- $1 = 1$ is already a binary decimal.
- $14\,641 = 11 \times 11 \times 11 \times 11$.
- $12\,221 = 11 \times 11 \times 101$.
- $10\,110 = 10\,110$ is already a binary decimal.

,

E. Nearly Shortest Repeating Substring

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

You are given a string s of length n consisting of lowercase Latin characters. Find the length of the shortest string k such that several (possibly one) copies of k can be concatenated together to form a string with the same length as s and, at most, one different character.

More formally, find the length of the shortest string k such that $c = \underbrace{k + \cdots + k}_{x \text{ times}}$ for some *positive integer* x, strings s and c has the same length and $c_i \neq s_i$ for at most one i (i.e. there exist 0 or 1 such positions).

Input

The first line contains a single integer t ($1 \leq t \leq 10^3$) — the number of test cases.

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

The second line of each test case contains the string s, consisting of lowercase Latin characters.

The sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print the length of the shortest string k satisfying the constraints in the statement.

Standard Input	Standard Output
5 4 abaa 4 abba 13 slavicgslavic 8 hshahaha 20 stormflamestornflame	1 4 13 2 10

Note

In the first test case, you can select $k = \texttt{a}$ and $k + k + k + k = \texttt{aaaa}$, which only differs from s in the second position.

In the second test case, you cannot select k of length one or two. We can have $k = \texttt{abba}$, which is equal to s.

,

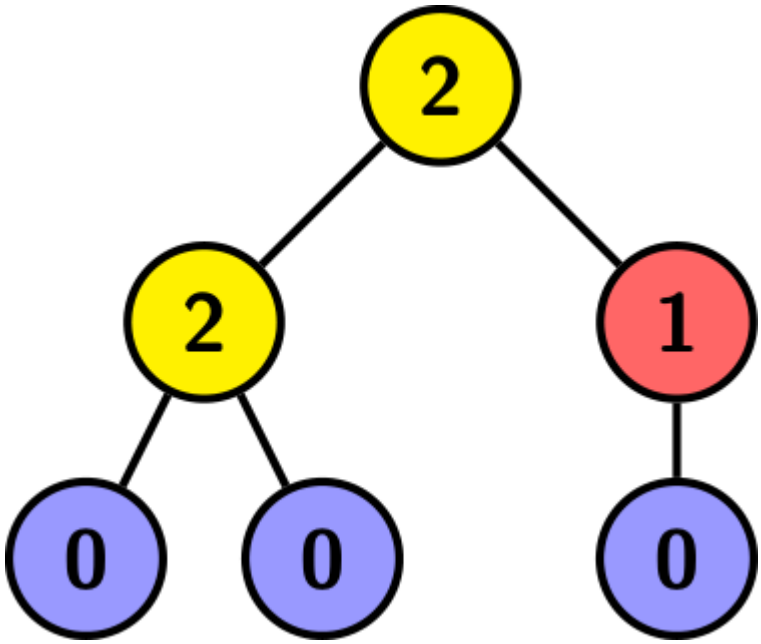
F. 0, 1, 2, Tree!

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

Find the minimum height of a rooted tree† with $a + b + c$ vertices that satisfies the following conditions:

- a vertices have exactly 2 children,
- b vertices have exactly 1 child, and
- c vertices have exactly 0 children.

If no such tree exists, you should report it.



The tree above is rooted at the top vertex, and each vertex is labeled with the number of children it has. Here $a = 2$, $b = 1$, $c = 3$, and the height is 2.

† A *rooted tree* is a connected graph without cycles, with a special vertex called the *root*. In a rooted tree, among any two vertices connected by an edge, one vertex is a parent (the one closer to the root), and the other one is a child.

The *distance* between two vertices in a tree is the number of edges in the shortest path between them. The *height* of a rooted tree is the maximum distance from a vertex to the root.

Input

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

The only line of each test case contains three integers a, b, and c ($0 \le a, b, c \le 10^5$; $1 \le a + b + c$).

The sum of $a + b + c$ over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, if no such tree exists, output -1. Otherwise, output one integer — the minimum height of a tree satisfying the conditions in the statement.

Standard Input	Standard Output

10 2 1 3 0 0 1 0 1 1 1 0 2 1 1 3 3 1 4 8 17 9 24 36 48 1 0 0 0 3 1	2 0 1 1 -1 3 6 -1 -1 3

Note

The first test case is pictured in the statement. It can be proven that you can't get a height smaller than 2.

In the second test case, you can form a tree with a single vertex and no edges. It has height 0, which is clearly optimal.

In the third test case, you can form a tree with two vertices joined by a single edge. It has height 1, which is clearly optimal.

,

G. Shuffling Songs

Input file:	standard input
Output file:	standard output
Time limit:	3 seconds
Memory limit:	256 megabytes

Vladislav has a playlist consisting of n songs, numbered from 1 to n. Song i has genre g_i and writer w_i. He wants to make a playlist in such a way that every pair of adjacent songs either have the same writer or are from the same genre (or both). He calls such a playlist *exciting*. Both g_i and w_i are strings of length no more than 10^4.

It might not always be possible to make an exciting playlist using all the songs, so the shuffling process occurs in two steps. First, some amount (possibly zero) of the songs are removed, and then the remaining songs in the playlist are rearranged to make it exciting.

Since Vladislav doesn't like when songs get removed from his playlist, he wants the making playlist to perform as few removals as possible. Help him find the minimum number of removals that need to be performed in order to be able to rearrange the rest of the songs to make the playlist exciting.

Input

The first line of the input contains a single integer t ($1 \le t \le 1000$) — the number of test cases. The description of test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 16$) — the number of songs in the original playlist.

Then n lines follow, the i-th of which contains two strings of lowercase letters g_i and w_i ($1 \le |g_i|, |w_i| \le 10^4$) — the genre and the writer of the i-th song. Where $|g_i|$ and $|w_i|$ are lengths of the strings.

The sum of 2^n over all test cases does not exceed 2^{16}.

The sum of $|g_i| + |w_i|$ over all test cases does not exceed $4 \cdot 10^5$.

Output

For each test case, output a single integer — the minimum number of removals necessary so that the resulting playlist can be made exciting.

Standard Input	Standard Output
4 1 pop taylorswift 4 electronic themotans electronic carlasdreams pop themotans pop irinarimes 7 rap eminem rap drdre rap kanyewest	0 0 4 3

pop taylorswift indierock arcticmonkeys indierock arcticmonkeys punkrock theoffspring 4 a b c d e f g h	

Note

In the first test case, the playlist is already exciting.

In the second test case, if you have the songs in the order $4, 3, 1, 2$, it is exciting, so you don't need to remove any songs.

In the third test case, you can remove songs $4, 5, 6, 7$. Then the playlist with songs in the order $1, 2, 3$ is exciting.