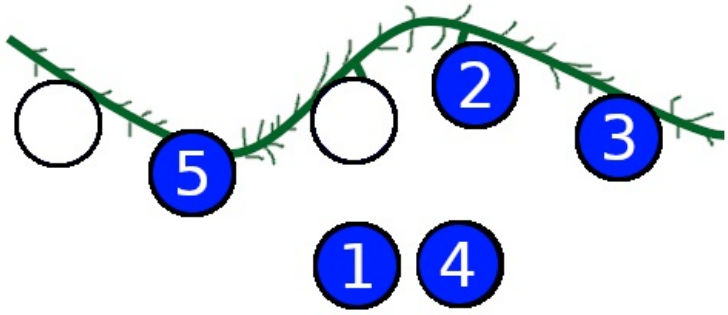
Codeforces Round #612 (Div. 1)

A. Garland

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

Vadim loves decorating the Christmas tree, so he got a beautiful garland as a present. It consists of n light bulbs in a single row. Each bulb has a number from 1 to n (in arbitrary order), such that all the numbers are distinct. While Vadim was solving problems, his home Carp removed some light bulbs from the garland. Now Vadim wants to put them back on.



Vadim wants to put all bulb back on the garland. Vadim defines *complexity* of a garland to be the number of pairs of adjacent bulbs with numbers with different parity (remainder of the division by 2). For example, the complexity of $1\ \ 4\ \ 2\ \ 3\ \ 5$ is 2 and the complexity of $1\ \ 3\ \ 5\ \ 7\ \ 6\ \ 4\ \ 2$ is 1.

No one likes complexity, so Vadim wants to minimize the number of such pairs. Find the way to put all bulbs back on the garland, such that the complexity is as small as possible.

Input

The first line contains a single integer n ($1 \le n \le 100$) — the number of light bulbs on the garland.

The second line contains n integers $p_1,\ p_2,\ \ldots,\ p_n$ ($0 \le p_i \le n$) — the number on the i-th bulb, or 0 if it was removed.

Output

Output a single number — the minimum *complexity* of the garland.

Examples

input
5 0 5 0 2 3
output
2

input
7 1 0 0 5 0 0 2
output
1

Note

In the first example, one should place light bulbs as $1\ \ 5\ \ 4\ \ 2\ \ 3$. In that case, the complexity would be equal to 2, because only $(5, 4)$ and $(2, 3)$ are the pairs of adjacent bulbs that have different parity.

In the second case, one of the correct answers is $1\ \ 7\ \ 3\ \ 5\ \ 6\ \ 4\ \ 2$.

B. Numbers on Tree

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

Evlampiy was gifted a rooted tree. The vertices of the tree are numbered from 1 to n. Each of its vertices also has an integer a_i written on it. For each vertex i, Evlampiy calculated c_i — the number of vertices j in the subtree of vertex i, such that $a_j < a_i$.

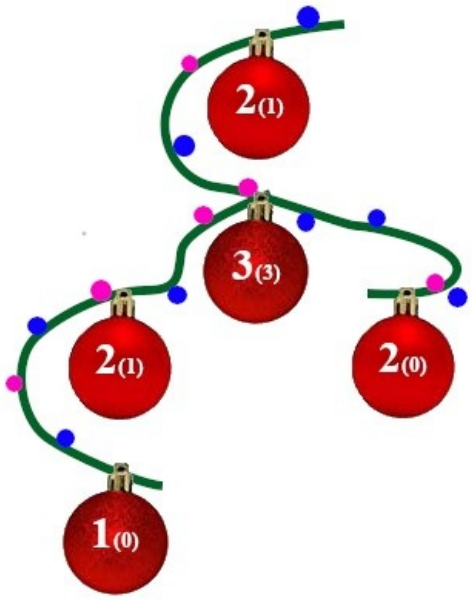


Illustration for the second example, the first integer is a_i and the integer in parentheses is c_i

After the new year, Evlampiy could not remember what his gift was! He remembers the tree and the values of c_i, but he completely forgot which integers a_i were written on the vertices.

Help him to restore initial integers!

Input

The first line contains an integer n $(1 \le n \le 2000)$ — the number of vertices in the tree.

The next n lines contain descriptions of vertices: the i-th line contains two integers p_i and c_i ($0 \le p_i \le n$; $0 \le c_i \le n-1$), where p_i is the parent of vertex i or 0 if vertex i is root, and c_i is the number of vertices j in the subtree of vertex i, such that $a_j < a_i$.

It is guaranteed that the values of p_i describe a rooted tree with n vertices.

Output

If a solution exists, in the first line print "YES", and in the second line output n integers a_i $(1 \le a_i \le 10^9)$. If there are several solutions, output any of them. One can prove that if there is a solution, then there is also a solution in which all a_i are between 1 and 10^9.

If there are no solutions, print "NO".

Examples

input
3 2 0 0 2 2 0
output
YES 1 2 1

input
5 0 1 1 3 2 1 3 0 2 0
output
YES 2 3 2 1 2

C1. Madhouse (Easy version)

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

This problem is different with hard version only by constraints on total answers length

It is an interactive problem

Venya joined a tour to the madhouse, in which orderlies play with patients the following game. Orderlies pick a string s of length n,

consisting only of lowercase English letters. The player can ask two types of queries:

- `? l r` – ask to list all substrings of $s[l..r]$. Substrings will be returned in random order, and in every substring, all characters will be randomly shuffled.
- `! s` – guess the string picked by the orderlies. This query can be asked exactly once, after that the game will finish. If the string is guessed correctly, the player wins, otherwise he loses.

The player can ask **no more than 3 queries** of the first type.

To make it easier for the orderlies, there is an additional limitation: the total number of returned substrings in all queries of the first type must not exceed $(n+1)^2$.

Venya asked you to write a program, which will guess the string by interacting with the orderlies' program and acting by the game's rules.

Your program should immediately terminate after guessing the string using a query of the second type. In case your program guessed the string incorrectly, or it violated the game rules, it will receive verdict `Wrong answer`.

Note that in every test case the string is fixed beforehand and will not change during the game, which means that the interactor is not adaptive.

Input
First line contains number n ($1 \le n \le 100$) — the length of the picked string.

Interaction
You start the interaction by reading the number n.

To ask a query about a substring from l to r inclusively ($1 \le l \le r \le n$), you should output

```
? l r
```

on a separate line. After this, all substrings of $s[l..r]$ will be returned in random order, each substring exactly once. In every returned substring all characters will be randomly shuffled.

In the case, if you ask an incorrect query, ask more than 3 queries of the first type or there will be more than $(n+1)^2$ substrings returned in total, you will receive verdict **Wrong answer**.

To guess the string s, you should output

```
! s
```

on a separate line.

After printing each query, do not forget to flush the output. Otherwise, you will get `Idleness limit exceeded`. To flush the output, you can use:

- fflush(stdout) or cout.flush() in C++;
- System.out.flush() in Java;
- flush(output) in Pascal;
- stdout.flush() in Python;
- see documentation for other languages.

If you received `-` (dash) as an answer to any query, you need to terminate your program with exit code 0 (for example, by calling `exit(0)`). This means that there was an error in the interaction protocol. If you don't terminate with exit code 0, you can receive any unsuccessful verdict.

Hack format

To hack a solution, use the following format:

The first line should contain one integer n ($1 \le n \le 100$) — the length of the string, and the following line should contain the string s.

Example

input
4 a aa a cb b c c
output
? 1 2 ? 3 4

```
? 4 4

! aabc
```

C2. Madhouse (Hard version)

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

This problem is different with easy version only by constraints on total answers length

It is an interactive problem

Venya joined a tour to the madhouse, in which orderlies play with patients the following game. Orderlies pick a string s of length n, consisting only of lowercase English letters. The player can ask two types of queries:

- `? l r` – ask to list all substrings of $s[l..r]$. Substrings will be returned in random order, and in every substring, all characters will be randomly shuffled.
- `! s` – guess the string picked by the orderlies. This query can be asked exactly once, after that the game will finish. If the string is guessed correctly, the player wins, otherwise he loses.

The player can ask **no more than 3 queries** of the first type.

To make it easier for the orderlies, there is an additional limitation: the total number of returned substrings in all queries of the first type must not exceed $\lceil 0.777(n+1)^2 \rceil$ ($\lceil x \rceil$ is x rounded up).

Venya asked you to write a program, which will guess the string by interacting with the orderlies' program and acting by the game's rules.

Your program should immediately terminate after guessing the string using a query of the second type. In case your program guessed the string incorrectly, or it violated the game rules, it will receive verdict `Wrong answer`.

Note that in every test case the string is fixed beforehand and will not change during the game, which means that the interactor is not adaptive.

Input

First line contains number n ($1 \le n \le 100$) — the length of the picked string.

Interaction

You start the interaction by reading the number n.

To ask a query about a substring from l to r inclusively ($1 \le l \le r \le n$), you should output

```
? l r
```

on a separate line. After this, all substrings of $s[l..r]$ will be returned in random order, each substring exactly once. In every returned substring all characters will be randomly shuffled.

In the case, if you ask an incorrect query, ask more than 3 queries of the first type or there will be more than $\lceil 0.777(n+1)^2 \rceil$ substrings returned in total, you will receive verdict **Wrong answer**.

To guess the string s, you should output

```
! s
```

on a separate line.

After printing each query, do not forget to flush the output. Otherwise, you will get `Idleness limit exceeded`. To flush the output, you can use:

- fflush(stdout) or cout.flush() in C++;
- System.out.flush() in Java;
- flush(output) in Pascal;
- stdout.flush() in Python;
- see documentation for other languages.

If you received `-` (dash) as an answer to any query, you need to terminate your program with exit code 0 (for example, by calling `exit(0)`). This means that there was an error in the interaction protocol. If you don't terminate with exit code 0, you can receive any unsuccessful verdict.

Hack format

To hack a solution, use the following format:

The first line should contain one integer n ($1 \le n \le 100$) — the length of the string, and the following line should contain the string s.

Example

input
4 a aa a cb b c c
output
? 1 2 ? 3 4 ? 4 4 ! aabc

D. LCC

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

An infinitely long Line Chillland Collider (LCC) was built in Chillland. There are n pipes with coordinates x_i that are connected to LCC. When the experiment starts at time 0, i-th proton flies from the i-th pipe with speed v_i. It flies to the right with probability p_i and flies to the left with probability $(1 - p_i)$. The *duration of the experiment* is determined as the time of the first collision of any two protons. In case there is no collision, the duration of the experiment is considered to be zero.

Find the expected value of the duration of the experiment.

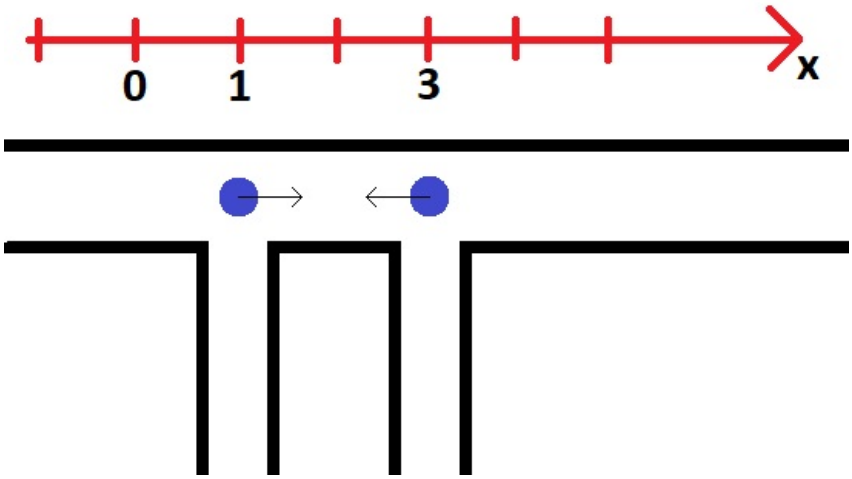


Illustration for the first example

Input

The first line of input contains one integer n — the number of pipes ($1 \le n \le 10^5$). Each of the following n lines contains three integers x_i, v_i, p_i — the coordinate of the i-th pipe, the speed of the i-th proton and the probability that the i-th proton flies to the right in percentage points ($-10^9 \le x_i \le 10^9, 1 \le v \le 10^6, 0 \le p_i \le 100$). It is guaranteed that all x_i are distinct and sorted in increasing order.

Output

It's possible to prove that the answer can always be represented as a fraction P/Q, where P is an integer and Q is a natural number not divisible by $998\,244\,353$. In this case, print $P \cdot Q^{-1}$ modulo $998\,244\,353$.

Examples

input
2 1 1 100 3 1 0
output
1

input
3 7 10 0

9 4 86
14 5 100

output

0

input

4
6 4 50
11 25 50
13 16 50
15 8 50

output

150902884

E. Fedya the Potter Strikes Back

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Fedya has a string S, initially empty, and an array W, also initially empty.

There are n queries to process, one at a time. Query i consists of a lowercase English letter c_i and a nonnegative integer w_i. First, c_i must be appended to S, and w_i must be appended to W. The answer to the query is the sum of *suspiciousnesses* for all subsegments of W $[L,\ R]$, $(1 \le L \le R \le i)$.

We define the *suspiciousness* of a subsegment as follows: if the substring of S corresponding to this subsegment (that is, a string of consecutive characters from L-th to R-th, inclusive) matches the prefix of S of the same length (that is, a substring corresponding to the subsegment $[1,\ R - L + 1]$), then its suspiciousness is equal to the minimum in the array W on the $[L,\ R]$ subsegment. Otherwise, in case the substring does not match the corresponding prefix, the suspiciousness is 0.

Help Fedya answer all the queries before the orderlies come for him!

Input

The first line contains an integer n $(1 \le n \le 600\,000)$ — the number of queries.

The i-th of the following n lines contains the query i: a lowercase letter of the Latin alphabet c_i and an integer w_i $(0 \le w_i \le 2^{30} - 1)$.

All queries are given in an encrypted form. Let ans be the answer to the previous query (for the first query we set this value equal to 0). Then, in order to get the real query, you need to do the following: perform a cyclic shift of c_i in the alphabet forward by ans, and set w_i equal to $w_i \oplus (ans\ \&\ MASK)$, where $\oplus$ is the bitwise exclusive "or", $\&$ is the bitwise "and", and $MASK = 2^{30} - 1$.

Output

Print n lines, i-th line should contain a single integer — the answer to the i-th query.

Examples

input

7
a 1
a 0
y 3
y 5
v 4
u 6
r 8

output

1
2
4
5
7
9
12

input

4
a 2
y 2
z 0
y 2

output

2
2
2

2

input
5 a 7 u 5 t 3 s 10 s 11
output
7 9 11 12 13

Note

For convenience, we will call "suspicious" those subsegments for which the corresponding lines are prefixes of S, that is, those whose suspiciousness may not be zero.

As a result of decryption in the first example, after all requests, the string S is equal to "abacaba", and all $w_i = 1$, that is, the suspiciousness of all suspicious sub-segments is simply equal to 1. Let's see how the answer is obtained after each request:

1. S = "a", the array W has a single subsegment — $[1,\ 1]$, and the corresponding substring is "a", that is, the entire string S, thus it is a prefix of S, and the suspiciousness of the subsegment is 1.

2. S = "ab", suspicious subsegments: $[1,\ 1]$ and $[1,\ 2]$, total 2.

3. S = "aba", suspicious subsegments: $[1,\ 1]$, $[1,\ 2]$, $[1,\ 3]$ and $[3,\ 3]$, total 4.

4. S = "abac", suspicious subsegments: $[1,\ 1]$, $[1,\ 2]$, $[1,\ 3]$, $[1,\ 4]$ and $[3,\ 3]$, total 5.

5. S = "abaca", suspicious subsegments: $[1,\ 1]$, $[1,\ 2]$, $[1,\ 3]$, $[1,\ 4]$, $[1,\ 5]$, $[3,\ 3]$ and $[5,\ 5]$, total 7.

6. S = "abacab", suspicious subsegments: $[1,\ 1]$, $[1,\ 2]$, $[1,\ 3]$, $[1,\ 4]$, $[1,\ 5]$, $[1,\ 6]$, $[3,\ 3]$, $[5,\ 5]$ and $[5,\ 6]$, total 9.

7. S = "abacaba", suspicious subsegments: $[1,\ 1]$, $[1,\ 2]$, $[1,\ 3]$, $[1,\ 4]$, $[1,\ 5]$, $[1,\ 6]$, $[1,\ 7]$, $[3,\ 3]$, $[5,\ 5]$, $[5,\ 6]$, $[5,\ 7]$ and $[7,\ 7]$, total 12.

In the second example, after all requests S = "aaba", $W = [2, 0, 2, 0]$.

1. S = "a", suspicious subsegments: $[1,\ 1]$ (suspiciousness 2), totaling 2.

2. S = "aa", suspicious subsegments: $[1,\ 1]$ (2), $[1,\ 2]$ (0), $[2,\ 2]$ (0), totaling 2.

3. S = "aab", suspicious subsegments: $[1,\ 1]$ (2), $[1,\ 2]$ (0), $[1,\ 3]$ (0), $[2,\ 2]$ (0), totaling 2.

4. S = "aaba", suspicious subsegments: $[1,\ 1]$ (2), $[1,\ 2]$ (0), $[1,\ 3]$ (0), $[1,\ 4]$ (0), $[2,\ 2]$ (0), $[4,\ 4]$ (0), totaling 2.

In the third example, from the condition after all requests S = "abcde", $W = [7, 2, 10, 1, 7]$.

1. S = "a", suspicious subsegments: $[1,\ 1]$ (7), totaling 7.

2. S = "ab", suspicious subsegments: $[1,\ 1]$ (7), $[1,\ 2]$ (2), totaling 9.

3. S = "abc", suspicious subsegments: $[1,\ 1]$ (7), $[1,\ 2]$ (2), $[1,\ 3]$ (2), totaling 11.

4. S = "abcd", suspicious subsegments: $[1,\ 1]$ (7), $[1,\ 2]$ (2), $[1,\ 3]$ (2), $[1,\ 4]$ (1), totaling 12.

5. S = "abcde", suspicious subsegments: $[1,\ 1]$ (7), $[1,\ 2]$ (2), $[1,\ 3]$ (2), $[1,\ 4]$ (1), $[1,\ 5]$ (1), totaling 13.

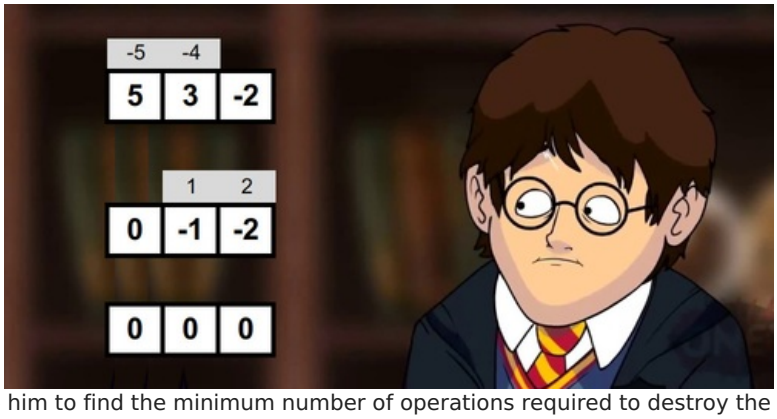
F. Harry The Potter

time limit per test: 9 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

To defeat Lord Voldemort, Harry needs to destroy all horcruxes first. The last horcrux is an array a of n integers, which also needs to be destroyed. The array is considered destroyed if all its elements are zeroes. To destroy the array, Harry can perform two types of operations:

1. choose an index i ($1 \le i \le n$), an integer x, and subtract x from a_i.
2. choose two indices i and j ($1 \le i, j \le n; i \ne j$), an integer x, and subtract x from a_i and $x + 1$ from a_j.

Note that x does not have to be positive.



Harry is in a hurry, please help him to find the minimum number of operations required to destroy the array and exterminate Lord Voldemort.

Input

The first line contains a single integer n — the size of the array a ($1 \le n \le 20$).

The following line contains n integers $a_1, a_2, \ldots, a_n$ — array elements ($-10^{15} \le a_i \le 10^{15}$).

Output

Output a single integer — the minimum number of operations required to destroy the array a.

Examples

input
3 1 10 100
output
3

input
3 5 3 -2
output
2

input
1 0
output
0

Note

In the first example one can just apply the operation of the first kind three times.

In the second example, one can apply the operation of the second kind two times: first, choose $i = 2, j = 1, x = 4$, it transforms the array into $(0, -1, -2)$, and then choose $i = 3, j = 2, x = -2$ to destroy the array.

In the third example, there is nothing to be done, since the array is already destroyed.