A. Maximum in Table

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

An $n \times n$ table a is defined as follows:

- The first row and the first column contain ones, that is: $a_{i,1} = a_{1,i} = 1$ for all $i = 1, 2, ..., n$.
- Each of the remaining numbers in the table is equal to the sum of the number above it and the number to the left of it. In other words, the remaining elements are defined by the formula $a_{i,j} = a_{i-1,j} + a_{i,j-1}$.

These conditions define all the values in the table.

You are given a number n. You need to determine the maximum value in the $n \times n$ table defined by the rules above.

Input

The only line of input contains a positive integer n $(1 \le n \le 10)$ — the number of rows and columns of the table.

Output

Print a single line containing a positive integer m — the maximum value in the table.

Sample test(s)

input
```
1
```
output
```
1
```

input
```
5
```
output
```
70
```

Note

In the second test the rows of the table look as follows:

$$\{1, 1, 1, 1, 1\},$$
$$\{1, 2, 3, 4, 5\},$$
$$\{1, 3, 6, 10, 15\},$$
$$\{1, 4, 10, 20, 35\},$$
$$\{1, 5, 15, 35, 70\}.$$

B. Painting Pebbles

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

There are n piles of pebbles on the table, the i-th pile contains a_i pebbles. Your task is to paint each pebble using one of the k given colors so that for each color c and any two piles i and j the difference between the number of pebbles of color c in pile i and number of pebbles of color c in pile j is at most one.

In other words, let's say that $b_{i, c}$ is the number of pebbles of color c in the i-th pile. Then for any $1 \le c \le k$, $1 \le i, j \le n$ the following condition must be satisfied $|b_{i, c} - b_{j, c}| \le 1$. It isn't necessary to use all k colors: if color c hasn't been used in pile i, then $b_{i, c}$ is considered to be zero.

Input

The first line of the input contains positive integers n and k $(1 \le n, k \le 100)$, separated by a space — the number of piles and the number of colors respectively.

The second line contains n positive integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le 100)$ denoting number of pebbles in each of the piles.

Output

If there is no way to paint the pebbles satisfying the given condition, output "`NO`" (without quotes) .

Otherwise in the first line output "`YES`" (without quotes). Then n lines should follow, the i-th of them should contain a_i space-separated integers. j-th $(1 \le j \le a_i)$ of these integers should be equal to the color of the j-th pebble in the i-th pile. **If there are several possible answers, you may output any of them.**

Sample test(s)

input

```
4 4
1 2 3 4
```

output

```
YES
1
1 4
1 2 4
1 2 3 4
```

input

```
5 2
3 2 4 1 3
```

output

```
NO
```

input

```
5 4
3 2 4 3 5
```

output

```
YES
1 2 3
1 3
1 2 3 4
1 3 4
1 1 2 3 4
```

C. Sums of Digits

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

Vasya had a **strictly increasing** sequence of positive integers a_1, ..., a_n. Vasya used it to build a new sequence b_1, ..., b_n, where b_i is the sum of digits of a_i's decimal representation. Then sequence a_i got lost and all that remained is sequence b_i.

Vasya wonders what the numbers a_i could be like. Of all the possible options he likes the one sequence with the minimum possible last number a_n. Help Vasya restore the initial sequence.

It is guaranteed that such a sequence always exists.

Input

The first line contains a single integer number n ($1 \le n \le 300$).

Next n lines contain integer numbers b_1, ..., b_n — the required sums of digits. All b_i belong to the range $1 \le b_i \le 300$.

Output

Print n integer numbers, one per line — the correct option for numbers a_i, in order of following in sequence. The sequence should be strictly increasing. The sum of digits of the i-th number should be equal to b_i.

If there are multiple sequences with least possible number a_n, print any of them. Print the numbers without leading zeroes.

Sample test(s)

input

```
3
1
2
3
```

output

```
1
2
3
```

input

```
3
3
2
1
```

output

```
3
11
100
```

D. Restoring Numbers

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

Vasya had two arrays consisting of non-negative integers: a of size n and b of size m. Vasya chose a positive integer k and created an $n \times m$ matrix v using the following formula:

$$v_{i,j} = (a_i + b_j) \bmod k$$

Vasya wrote down matrix v on a piece of paper and put it in the table.

A year later Vasya was cleaning his table when he found a piece of paper containing an $n \times m$ matrix w. He remembered making a matrix one day by the rules given above but he was not sure if he had found the paper with the matrix v from those days. Your task is to find out if the matrix w that you've found could have been obtained by following these rules and if it could, then for what numbers $k, a_1, a_2, ..., a_n, b_1, b_2, ..., b_m$ it is possible.

Input

The first line contains integers n and m ($1 \le n, m \le 100$), separated by a space — the number of rows and columns in the found matrix, respectively.

The i-th of the following lines contains numbers $w_{i,1}, w_{i,2}, ..., w_{i,m}$ ($0 \le w_{i,j} \le 10^9$), separated by spaces — the elements of the i-th row of matrix w.

Output

If the matrix w could not have been obtained in the manner described above, print "`NO`" (without quotes) in the single line of output.

Otherwise, print four lines.

In the first line print "`YES`" (without quotes).

In the second line print an integer k ($1 \le k \le 10^{18}$). Note that each element of table w should be in range between 0 and $k - 1$ inclusively.

In the third line print n integers $a_1, a_2, ..., a_n$ ($0 \le a_i \le 10^{18}$), separated by spaces.

In the fourth line print m integers $b_1, b_2, ..., b_m$ ($0 \le b_i \le 10^{18}$), separated by spaces.

Sample test(s)

input
```
2 3
1 2 3
2 3 4
```
output
```
YES
1000000007
0 1
1 2 3
```

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

input
```
2 2
1 2
2 1
```
output
```
NO
```

Note

By $b \bmod c$ we denote the remainder of integer division of b by c.

It is guaranteed that if there exists some set of numbers $k, a_1, ..., a_n, b_1, ..., b_m$, that you could use to make matrix w, then there also exists a set of numbers that meets the limits $1 \le k \le 10^{18}$, $1 \le a_i \le 10^{18}$, $1 \le b_i \le 10^{18}$ in the output format. In other words, these upper bounds are introduced only for checking convenience purposes.

E. Pretty Song

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

When Sasha was studying in the seventh grade, he started listening to music a lot. In order to evaluate which songs he likes more, he introduced the notion of the song's *prettiness*. The title of the song is a word consisting of uppercase Latin letters. The *prettiness* of the song is the *prettiness* of its title.

Let's define the *simple prettiness* of a word as the ratio of the number of vowels in the word to the number of all letters in the word.

Let's define the *prettiness* of a word as the sum of *simple prettiness* of all the substrings of the word.

More formally, let's define the function $vowel(c)$ which is equal to 1, if c is a vowel, and to 0 otherwise. Let s_i be the i-th character of string s, and $s_{i..j}$ be the substring of word s, staring at the i-th character and ending at the j-th character ($s_is_{i+1}\dots s_j$, $i \le j$).

Then the *simple prettiness* of s is defined by the formula:

$$simple(s) = \frac{\sum_{i=1}^{|s|} vowel(s_i)}{|s|}$$

The *prettiness* of s equals

$$\sum_{1 \le i \le j \le |s|} simple(s_{i..j}).$$

Find the *prettiness* of the given song title.

We assume that the vowels are I, E, A, O, U, Y.

Input

The input contains a single string s ($1 \le |s| \le 5 \cdot 10^5$) — the title of the song.

Output

Print the *prettiness* of the song with the absolute or relative error of at most 10^{-6}.

Sample test(s)

input
```
IEAIAIO
```
output
```
28.0000000
```

input
```
BYOB
```
output
```
5.8333333
```

input
```
YISVOWEL
```
output
```
17.0500000
```

Note

In the first sample all letters are vowels. The *simple prettiness* of each substring is 1. The word of length 7 has 28 substrings. So, the *prettiness* of the song equals to 28.

F. Progress Monitoring

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

Programming teacher Dmitry Olegovich is going to propose the following task for one of his tests for students:

You are given a tree T with n vertices, specified by its adjacency matrix $a[1... n, 1... n]$. What is the output of the following pseudocode?

```
used[1 ... n] = {0, ..., 0};

procedure dfs(v):
    print v;
    used[v] = 1;
    for i = 1, 2, ..., n:
        if (a[v][i] == 1 and used[i] == 0):
            dfs(i);

dfs(1);
```

In order to simplify the test results checking procedure, Dmitry Olegovich decided to create a tree T such that the result is his favorite sequence b. On the other hand, Dmitry Olegovich doesn't want to provide students with same trees as input, otherwise they might cheat. That's why Dmitry Olegovich is trying to find out the number of different trees T such that the result of running the above pseudocode with T as input is exactly the sequence b. Can you help him?

Two trees with n vertices are called different if their adjacency matrices a_1 and a_2 are different, i. e. there exists a pair (i, j), such that $1 \le i, j \le n$ and $a_1[i][j] \ne a_2[i][j]$.

Input

The first line contains the positive integer n $(1 \le n \le 500)$ — the length of sequence b.

The second line contains n positive integers $b_1, b_2, ..., b_n$ $(1 \le b_i \le n)$. It is guaranteed that b is a permutation, or in other words, each of the numbers $1, 2, ..., n$ appears exactly once in the sequence b. Also it is guaranteed that $b_1 = 1$.

Output

Output the number of trees satisfying the conditions above modulo $10^9 + 7$.

Sample test(s)

input

```
3
1 2 3
```

output

```
2
```

input

```
3
1 3 2
```

output

```
1
```