Problem B: Binary Search Tree

In Tobyland, dogs communicate with each other through barking. These barks are like symbols in a complex language, and naturally different symbols have different probabilities of appearing. Toby enumerates the symbols (barks) from 1 to N in the dog alphabetic order, so symbol 1 is lexicographically smaller than 2 and so on. Toby also counts in many dog conversations the frequency of each symbol to estimate its probabilities.

Now Toby wants to put these symbols (barks) in a fast data structure (Toby is the only dog that studies computer science), and he chose a binary search tree (BST). As in any BST, all nodes that are to the left of some node with symbol X must be smaller lexicographically than X and nodes to the right must have symbols bigger than X. Toby is incredibly concerned with efficiency; he defines the cost of searching a symbol as the number of nodes that have to be visited from the root to the node with the desired symbol—if the searched symbol is at the root the cost is 1. Now Toby wants to find the BST that minimizes the expected cost of searching a symbol.

Input

The input consist of several test cases. Each test case begins with a line containing the number N, which is the number of symbols in the dog alphabet, the second line contains N numbers p_i stating the probability of receiving the symbol i. It is guaranteed that the sum of the probabilities is 1.

$$1 \leqslant N \leqslant 100 \;\; ; \;\; 0 \leqslant p_i \leqslant 1 \;\; ; \;\; \sum_{i=1}^{N} p_i = 1$$

Output

For each test case write the minimum expected cost of searching a symbol in the BST. Print one line per test case. Answers with a relative or absolute error less than 10^{-4} are considered correct.

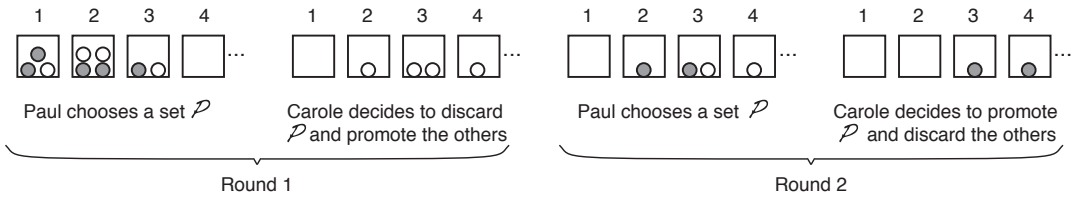
Sample Input	Output for Sample Input
3 0.33 0.34 0.33 3 0.8 0.15 0.05 4 0.23 0.4 0.17 0.2	1.6600 1.2500 1.7700

Problem B

Boxes and Stones

Paul and Carole like to play a game with S stones and B boxes numbered from 1 to B. Before beginning the game they arbitrarily distribute the S stones among the boxes from 1 to $B-1$, leaving box B empty. The game then proceeds by rounds. At each round, first Paul chooses a subset $\mathcal{P}$ of the stones that are in the boxes; he may choose as many stones as he wants from as many boxes as he wants, or he may choose no stones at all, in which case $\mathcal{P}$ is empty. Then, Carole decides what to do next: she can either *promote* the subset $\mathcal{P}$ and *discard* the remaining stones (that is, those stones not chosen by Paul in the first step); or she may *discard* the subset $\mathcal{P}$ and *promote* the remaining stones.

To *promote* a given subset means to take each stone in this subset and move it to the box with the next number in sequence, so that if there was a stone in this subset inside box b, it is moved to box $b+1$. To *discard* a given subset means to remove every stone in this subset from its corresponding box, so that those stones are not used in the game for the remaining rounds. The figure below shows an example of the first two rounds of a game.



Paul and Carole play until at least one stone reaches box number B, in which case Paul wins the game, or until there are no more stones left in the boxes, in which case Carole wins the game. Paul is a very rational player, but Carole is a worthy rival because she is not only extremely good at this game, but also quite lucky. We would like to know who is the best player, but before that we must first understand how the outcome of a game depends on the initial distribution of the stones. In particular, we would like to know in how many ways the S stones can initially be distributed among the first $B-1$ boxes so that Carole can be certain that she can win the game if she plays optimally, even if Paul never makes a mistake.

Input

Each test case is described using one line. The line contains two integers S ($1 \leq S \leq 200$) and B ($2 \leq B \leq 100$), representing respectively the number of stones and the number of boxes in the game.

Output

For each test case output a line with an integer representing the number of ways in which the S stones may be distributed among the first $B-1$ boxes so that Carole is certain that she can win the game. Because this number can be very large, you are required to output the remainder of dividing it by 10^9+7.

Sample input	**Output for the sample input**
2 3 8 4 42 42	2 0 498467348

Problem C: Sumthing

Has it ever happened to you that, having worked on a problem for a long time, it starts to pop up in your conscious mind when you least expect it? Just the other day I was singing that old song that goes *"Something in the way she moves..."*, but before I knew it, I replaced part of the lyrics with *"Sum-thing in the way she woos me..."*. The only explanation I have for this is that I had been working recently on a curious mathematical problem concerning sums. It goes something like this:

Consider a list A with n positive integers, $A_1, A_2, A_3, \ldots, A_n$. A function S is defined as follows, for $1 \leqslant k \leqslant n$:

$$S(k) = 2^{k-1} \sum_{i_1=1}^{n} \sum_{i_2=i_1+1}^{n} \sum_{i_3=i_2+1}^{n} \cdots \sum_{i_k=i_{k-1}+1}^{n} A_{i_1} A_{i_2} A_{i_3} \cdots A_{i_k}$$

For example, if $A = (1, 2, 3)$, then the possible values of S are:

$$S(1) = 1 + 2 + 3 = 6$$
$$S(2) = 2 \cdot ((1 \cdot 2) + (1 \cdot 3) + (2 \cdot 3)) = 2(2 + 3 + 6) = 22$$
$$S(3) = 4 \cdot (1 \cdot 2 \cdot 3) = 4 \cdot 6 = 24$$

What the problem asks is, given the list A, find the sum:

$$\Phi = \sum_{k=1}^{n} S(k)$$

Input

Input starts with an integer T, the number of test cases. Each test case starts with an integer n in the first line. The second line of each case contains n positive integers, separated by spaces, that form the set A.

$T \leqslant 10$; $1 \leqslant n \leqslant 10^5$; $1 \leqslant A_i \leqslant 10^9$ for $1 \leqslant i \leqslant n$

Output

For each test case, print the value of Φ, modulo 1000000009 $(10^9 + 9)$ on a single line.

Sample Input	Output for Sample Input
2 3 1 2 3 5 2 3 5 7 11	52 66412

Problem D
ICPC Strikes Again

Source file name: `icpc.c`, `icpc.cpp`, `icpc.java` *or* `icpc.pas`

International Concrete Projects Company (ICPC) is a construction company which specializes in building houses for the high-end market. The company is the most profitable company in the world due to a very efficient land division method which has been used in its housing development projects since last year. Recently there was a chaos at ICPC, because employees refused to work arguing that they did not earn enough. Worried about the loss in profit due to the strike, the company board proposed a new method to calculate the salaries which was luckily accepted by everyone.

The salary of a worker reflects the significance of the tasks that he/she has to perform and is influenced by the way tasks depend on each other.

A task X *depends* on a task Y if either (i) X *depends directly* on Y, or (ii) there exists a task T such that X depends directly on T and T depends on Y. Since in ICPC all tasks must be performed, there is no circularity in the task dependence relation. Also, a task may be performed by more than one worker.

A *basic significance* is associated with each task reflecting its importance (for example, developing the efficient land division method is more important than building the houses themselves). The *significance* of a task T is then defined as the basic significance of T plus the significance of every task which depends directly on T. Note that if no other tasks depend directly on task T, the basic significance and the significance of T are the same.

The salary of a worker is the sum of the significances of all the tasks he/she performs which do not depend on any other task performed by him/her. In other words, a value equal to the significance of task X will be added to the salary of a worker W that works in task X if there is no other task Y on which X depends, and W works also in Y.

ICPC wants you to help them to determine the salary of each of its employees.

Input

The input contains several test cases.

The first line of a test case contains two integers T and E indicating respectively the number of tasks and the number of employees ($1 \leq T \leq 1000$ and $1 \leq E \leq 1000$). Tasks are numbered from 1 to T and employees from 1 to E.

Then it will come a sequence of lines describing the tasks 1 to T in ascending order. Each task is described by two lines. The first of these lines contains three integers BS, ND and NE, representing respectively the basic significance of the task, the number of tasks that depend directly on it, and the number of employees who perform it ($1 \leq BS \leq 1000$, $0 \leq ND < T$ and $1 \leq NE \leq E$). The second line contains $ND + NE$ integers corresponding first to the ND directly dependent tasks and then the NE employees who perform the task.

The end of input is indicated by $T = E = 0$.

The input must be read from standard input.

Output

Test cases must be answered in the order that they were presented. For each test case you must print:

- a single line containing five stars ***** indicating the beginning of the case
- for each employee i, one line with two integers i and s, separated by a blank, meaning that i has a salary of s.

The output must be written to standard output.

Sample input	**Output for the sample input**
3 2 100 2 2 2 3 1 2 40 0 1 1 60 0 1 2 7 2 10 2 1 2 3 1 10 2 1 4 5 2 10 2 1 6 7 2 10 0 1 1 10 0 1 1 10 0 1 1 10 0 1 1 0 0	***** 1 200 2 200 ***** 1 70 2 60

Problem E
Eleven

In this problem, we refer to the digits of a positive integer as the sequence of digits required to write it in base 10 without leading zeros. For instance, the digits of $N = 2090$ are of course 2, 0, 9 and 0.

Let N be a positive integer. We call a positive integer M an eleven-multiple-anagram of N if and only if (1) the digits of M are a permutation of the digits of N, and (2) M is a multiple of 11. You are required to write a program that given N, calculates the number of its eleven-multiple-anagrams.

As an example, consider again $N = 2090$. The values that meet the first condition above are 2009, 2090, 2900, 9002, 9020 and 9200. Among those, only 2090 and 9020 satisfy the second condition, so the answer for $N = 2090$ is 2.

Input

A single line that contains an integer N ($1 \leq N \leq 10^{100}$).

Output

Output a line with an integer representing the number of eleven-multiple-anagrams of N. Because this number can be very large, you are required to output the remainder of dividing it by $10^9 + 7$.

Sample input 1	**Sample output 1**
2090	2

Sample input 2	**Sample output 2**
16510	12

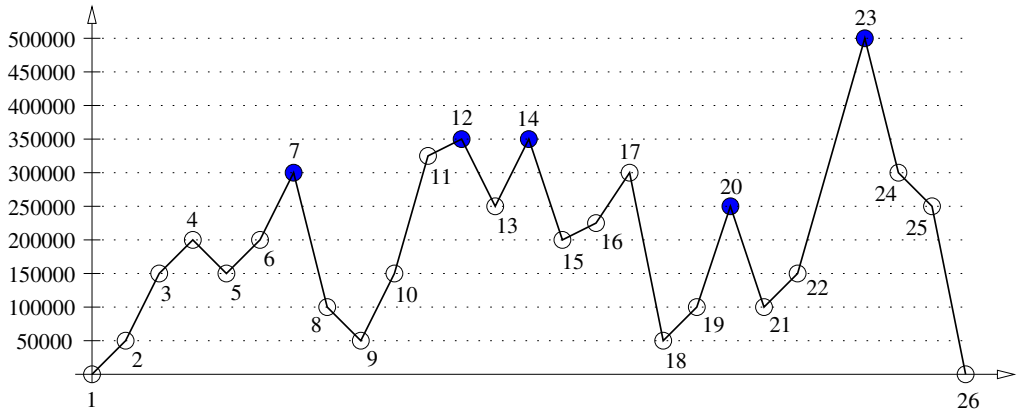
Sample input 3	**Sample output 3**
20140000000000000000000000000	0

Problem G

Go up the Ultras

The topographic prominence of a peak is a measure of special interest to mountain climbers and can be defined as follows: the prominence of a peak p with altitude h, relative to the sea level, is the greatest d such that any path on the terrain from p to any strictly higher peak will pass through a point of altitude $h - d$. If there is no strictly higher peak, then the prominence is h itself. Those peaks with topographic prominence greater than or equal to 150000 centimeters (precision is of great importance to climbers!) have a special name: they are called "Ultras".

You have to write a program that identifies all the Ultras that occur in a two dimensional profile of a mountain range represented as a sequence of points. Note that the horizontal distance between points is not important; all that you need is the altitude of each point. In the picture below, the Ultras are the points 7, 12, 14, 20 and 23.



Input

The first line contains an integer N ($3 \leq N \leq 10^5$) representing the number of points in the profile. The second line contains N integers H_i indicating the altitudes (in centimeters) of the points, in the order in which they appear in the profile ($0 \leq H_i \leq 10^6$ for $i = 1, 2, \ldots, N$). Consecutive points have different altitudes ($H_i \neq H_{i+1}$ for $i = 1, 2, \ldots, N - 1$), while the first and the last points are at sea level ($H_1 = H_N = 0$). You may assume that the profile contains at least one Ultra.

Output

Output a line with the indices of all the Ultras in the mountain range, in the order in which they appear in the profile.

Sample input 1	**Sample output 1**
5 0 10000 100000 884813 0	4

Sample input 2	**Sample output 2**
7 0 100000 0 200000 180000 200000 0	4 6

Problem J

Join two kingdoms

The kingdoms of Nlogonia and Quadradonia fought a long and terrible war that historians have come to call Almost Completely Meaningless (ACM) because nobody can now remember why it started. When the ACM war finally ended, the two kingdoms decided to strengthen their bonds in order to avoid more bloodshed, and for this reason they consulted the International Consortium for the Prevention of Conflicts (ICPC). The ICPC recommended building a single road to connect a city in Nlogonia with a city in Quadradonia, thus allowing commercial and cultural exchange between the two.

Nlogonia and Quadradonia have N and Q cities respectively. The road system of each kingdom consists of a set of bidirectional roads that join pairs of different cities in the same kingdom, such that there is a unique path (i.e. sequence of consecutive roads) that one can take to go from any city in a kingdom to any other city in the same kingdom. The "size" of such a road system is defined as the maximum number of roads that one must take in order to travel between any pair of cities.

Because the ICPC did not specify which two cities should be connected by the new road joining the two kingdoms, the citizens are now worried that the size of the combined road system might be too large. In order to prevent a second ACM war, you would like to convince them that this is not the case, and to this end you need to calculate the expected size of the resulting road system assuming that all possible roads between the two kingdoms are equally likely to be built.

Input

The first line contains two integers N and Q representing the number of cities in each of the two kingdoms ($1 \leq N, Q \leq 4 \times 10^4$). Cities in Nlogonia are identified with different integers from 1 to N, while cities in Quadradonia are identified with different integers from 1 to Q. Each of the next $N-1$ lines describes a road in Nlogonia with two distinct integers A and B indicating that the road connects city A with city B ($1 \leq A, B \leq N$). Each of the next $Q-1$ lines describes a road in Quadradonia with two distinct integers C and D indicating that the road connects city C with city D ($1 \leq C, D \leq Q$). The road system of each kingdom is such that there is exactly one path between each pair of cities in the kingdom.

Output

Output a line with a rational number representing the expected size of the road system after the two kingdoms have been joined, considering that all possible roads connecting them are equally likely to be built. The result must be output as a rational number with exactly three digits after the decimal point, rounded if necessary.

Sample input 1	**Sample output 1**
4 5 1 2 2 3 4 2 2 3 3 4 4 1 4 5	5.350

Sample input 2	**Sample output 2**
1 5 1 2 2 3 3 4 4 5	4.400

Problem K
Kids' Wishes

Problem code name: `kids`

Kevin is a kid. He has lunch at school together with many more kids. They use to go outdoors and have lunch sitting on the ground. They love to form a big circle in which each kid has exactly two neighbors, one on the left and one on the right. Sometimes the teacher has problems arranging the circle because some kids wish to sit down next to other kids. Each kid may wish to sit down next to at most two other kids, because each kid has just two neighbors in the circle. The teacher wants to know whether it is possible to arrange the circle in such a way that all kids' wishes are satisfied. You clean up the place when the lunch ends. Since you want to finish your work as early as possible, help the teacher in answering that question.

Input

Each test case is given using several lines. The first line contains two integers K and W representing respectively the number of kids ($3 \leq K \leq 10^9$) and the number of wishes ($0 \leq W \leq 10^5$). Kids are identified with numbers between 1 and K. Each of the next W lines describes a different wish using two distinct integers A and B ($1 \leq A, B \leq K$); these values represent that kid A wishes to sit down next to kid B. Each kid has at most two wishes.

The last test case is followed by a line containing two zeros.

Output

For each test case output a single line containing an uppercase '`Y`' if it is possible to arrange a circle in such a way that all kids' wishes are satisfied, or an uppercase '`N`' otherwise.

Sample input	**Output for the sample input**

`4 3`
`2 3`
`1 3`
`2 1`
`1000000000 0`
`3 6`
`3 2`
`2 1`
`1 2`
`1 3`
`2 3`
`3 1`
`0 0`
	`N`
	`Y`
	`Y`