Codeforces Round #513 by Barcelona Bootcamp (rated, Div. 1 + Div. 2)

A. Phone Numbers

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Let's call a string *a phone number* if it has length 11 and fits the pattern "8xxxxxxxxxx", where each "x" is replaced by a digit.

For example, "80123456789" and "80000000000" are phone numbers, while "8012345678" and "79000000000" are not.

You have n cards with digits, and you want to use them to make as many phone numbers as possible. Each card must be used in at most one phone number, and you don't have to use all cards. The phone numbers do not necessarily have to be distinct.

Input

The first line contains an integer n — the number of cards with digits that you have ($1 \leq n \leq 100$).

The second line contains a string of n digits (characters "0", "1", ..., "9") $s_1, s_2, \ldots, s_n$. The string will not contain any other characters, such as leading or trailing spaces.

Output

If at least one phone number can be made from these cards, output the maximum number of phone numbers that can be made. Otherwise, output 0.

Examples

input
```
11
00000000008
```
output
```
1
```

input
```
22
0011223344556677889988
```
output
```
2
```

input
```
11
31415926535
```
output
```
0
```

Note

In the first example, one phone number, "8000000000", can be made from these cards.

In the second example, you can make two phone numbers from the cards, for example, "80123456789" and "80123456789".

In the third example you can't make any phone number from the given cards.

B. Maximum Sum of Digits

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given a positive integer n.

Let $S(x)$ be sum of digits in base 10 representation of x, for example, $S(123) = 1 + 2 + 3 = 6$, $S(0) = 0$.

Your task is to find two integers a, b, such that $0 \leq a, b \leq n$, $a + b = n$ and $S(a) + S(b)$ is the largest possible among all such pairs.

Input

The only line of input contains an integer $$$n$$$ $$$(1 \leq n \leq 10^{12})$$$.

Output

Print largest $$$S(a) + S(b)$$$ among all pairs of integers $$$a, b$$$, such that $$$0 \leq a, b \leq n$$$ and $$$a + b = n$$$.

Examples

input

```
35
```

output

```
17
```

input

```
10000000000
```

output

```
91
```

Note

In the first example, you can choose, for example, $$$a = 17$$$ and $$$b = 18$$$, so that $$$S(17) + S(18) = 1 + 7 + 1 + 8 = 17$$$. It can be shown that it is impossible to get a larger answer.

In the second test example, you can choose, for example, $$$a = 5000000001$$$ and $$$b = 4999999999$$$, with $$$S(5000000001) + S(4999999999) = 91$$$. It can be shown that it is impossible to get a larger answer.

C. Maximum Subrectangle

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given two arrays $$$a$$$ and $$$b$$$ of positive integers, with length $$$n$$$ and $$$m$$$ respectively.

Let $$$c$$$ be an $$$n \times m$$$ matrix, where $$$c_{i,j} = a_i \cdot b_j$$$.

You need to find a subrectangle of the matrix $$$c$$$ such that the sum of its elements is at most $$$x$$$, and its area (the total number of elements) is the largest possible.

Formally, you need to find the largest number $$$s$$$ such that it is possible to choose integers $$$x_1, x_2, y_1, y_2$$$ subject to $$$1 \leq x_1 \leq x_2 \leq n$$$, $$$1 \leq y_1 \leq y_2 \leq m$$$, $$$(x_2 - x_1 + 1) \times (y_2 - y_1 + 1) = s$$$, and $$$$$$\sum_{i=x_1}^{x_2}{\sum_{j=y_1}^{y_2}{c_{i,j}}} \leq x.$$$$$$

Input

The first line contains two integers $$$n$$$ and $$$m$$$ ($$$1 \leq n, m \leq 2000$$$).

The second line contains $$$n$$$ integers $$$a_1, a_2, \ldots, a_n$$$ ($$$1 \leq a_i \leq 2000$$$).

The third line contains $$$m$$$ integers $$$b_1, b_2, \ldots, b_m$$$ ($$$1 \leq b_i \leq 2000$$$).

The fourth line contains a single integer $$$x$$$ ($$$1 \leq x \leq 2 \cdot 10^{9}$$$).

Output

If it is possible to choose four integers $$$x_1, x_2, y_1, y_2$$$ such that $$$1 \leq x_1 \leq x_2 \leq n$$$, $$$1 \leq y_1 \leq y_2 \leq m$$$, and $$$\sum_{i=x_1}^{x_2}{\sum_{j=y_1}^{y_2}{c_{i,j}}} \leq x$$$, output the largest value of $$$(x_2 - x_1 + 1) \times (y_2 - y_1 + 1)$$$ among all such quadruplets, otherwise output $$$0$$$.

Examples

input

```
3 3
1 2 3
1 2 3
9
```

output

```
4
```

input

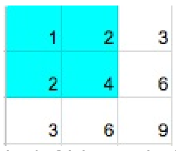
```
5 1
5 4 2 4 5
2
5
```

output

```
1
```

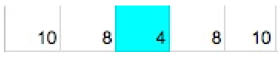
Note

Matrix from the first sample and the chosen subrectangle (of blue color):



1	2	3
2	4	6
3	6	9

Matrix from the second sample and the chosen subrectangle (of blue color):



10	8	4	8	10

D. Social Circles

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You invited $$$n$$$ guests to dinner! You plan to arrange one or more circles of chairs. Each chair is going to be either occupied by one guest, or be empty. You can make any number of circles.

Your guests happen to be a little bit shy, so the $$$i$$$-th guest wants to have a least $$$l_i$$$ free chairs to the left of his chair, and at least $$$r_i$$$ free chairs to the right. The "left" and "right" directions are chosen assuming all guests are going to be seated towards the center of the circle. Note that when a guest is the only one in his circle, the $$$l_i$$$ chairs to his left and $$$r_i$$$ chairs to his right may overlap.

What is smallest total number of chairs you have to use?

Input

First line contains one integer $$$n$$$ — number of guests, ($$$1 \leqslant n \leqslant 10^5$$$).

Next $$$n$$$ lines contain $$$n$$$ pairs of space-separated integers $$$l_i$$$ and $$$r_i$$$ ($$$0 \leqslant l_i, r_i \leqslant 10^9$$$).

Output

Output a single integer — the smallest number of chairs you have to use.

Examples

input
```
3
1 1
1 1
1 1
```
output
```
6
```

input
```
4
1 2
2 1
3 5
5 3
```
output
```
15
```

input
```
1
5 6
```
output
```
7
```

Note

In the second sample the only optimal answer is to use two circles: a circle with $$$5$$$ chairs accomodating guests $$$1$$$ and $$$2$$$, and another one with $$$10$$$ chairs accomodationg guests $$$3$$$ and $$$4$$$.

In the third sample, you have only one circle with one person. The guest should have at least five free chairs to his left, and at least six free chairs to his right to the next person, which is in this case the guest herself. So, overall number of chairs should be at least 6+1=7.

E. Sergey and Subway

time limit per test: 2 seconds
memory limit per test: 512 megabytes

input: standard input

output: standard output

Sergey Semyonovich is a mayor of a county city N and he used to spend his days and nights in thoughts of further improvements of Nkers' lives. Unfortunately for him, anything and everything has been done already, and there are no more possible improvements he can think of during the day (he now prefers to sleep at night). However, his assistants have found a solution and they now draw an imaginary city on a paper sheet and suggest the mayor can propose its improvements.

Right now he has a map of some imaginary city with $$$n$$$ subway stations. Some stations are directly connected with tunnels in such a way that the whole map is a tree (assistants were short on time and enthusiasm). It means that there exists exactly one simple path between each pair of station. We call a path simple if it uses each tunnel no more than once.

One of Sergey Semyonovich's favorite quality objectives is the sum of all pairwise distances between every pair of stations. The distance between two stations is the minimum possible number of tunnels on a path between them.

Sergey Semyonovich decided to add new tunnels to the subway map. In particular, he connected any two stations $$$u$$$ and $$$v$$$ that were not connected with a direct tunnel but share a common neighbor, i.e. there exists such a station $$$w$$$ that the original map has a tunnel between $$$u$$$ and $$$w$$$ and a tunnel between $$$w$$$ and $$$v$$$. You are given a task to compute the sum of pairwise distances between all pairs of stations in the new map.

Input

The first line of the input contains a single integer $$$n$$$ ($$$2 \leq n \leq 200\,000$$$) — the number of subway stations in the imaginary city drawn by mayor's assistants. Each of the following $$$n - 1$$$ lines contains two integers $$$u_i$$$ and $$$v_i$$$ ($$$1 \leq u_i, v_i \leq n$$$, $$$u_i \ne v_i$$$), meaning the station with these indices are connected with a direct tunnel.

It is guaranteed that these $$$n$$$ stations and $$$n - 1$$$ tunnels form a tree.

Output

Print one integer that is equal to the sum of distances between all pairs of stations after Sergey Semyonovich draws new tunnels between all pairs of stations that share a common neighbor in the original map.

Examples

input

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

output

```
6
```

input

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

output

```
7
```

Note

In the first sample, in the new map all pairs of stations share a direct connection, so the sum of distances is $$$6$$$.

In the second sample, the new map has a direct tunnel between all pairs of stations except for the pair $$$(1, 4)$$$. For these two stations the distance is $$$2$$$.

F. Shrinking Tree

time limit per test: 2 seconds

memory limit per test: 512 megabytes

input: standard input

output: standard output

Consider a tree $$$T$$$ (that is, a connected graph without cycles) with $$$n$$$ vertices labelled $$$1$$$ through $$$n$$$. We start the following process with $$$T$$$: while $$$T$$$ has more than one vertex, do the following:

- choose a random edge of $$$T$$$ equiprobably;
- *shrink* the chosen edge: if the edge was connecting vertices $$$v$$$ and $$$u$$$, erase both $$$v$$$ and $$$u$$$ and create a new vertex adjacent to all vertices previously adjacent to either $$$v$$$ or $$$u$$$. The new vertex is labelled either $$$v$$$ or $$$u$$$ equiprobably.

At the end of the process, $$$T$$$ consists of a single vertex labelled with one of the numbers $$$1, \ldots, n$$$. For each of the numbers, what is the probability of this number becoming the label of the final vertex?

Input

The first line contains a single integer $$$n$$$ ($$$1 \leq n \leq 50$$$).

The following $$$n - 1$$$ lines describe the tree edges. Each of these lines contains two integers $$$u_i, v_i$$$ — labels of vertices connected by the respective edge ($$$1 \leq u_i, v_i \leq n$$$, $$$u_i \neq v_i$$$). It is guaranteed that the given graph is a tree.

Output

Print $$$n$$$ floating numbers — the desired probabilities for labels $$$1, \ldots, n$$$ respectively. All numbers should be correct up to $$$10^{-6}$$$ relative or absolute precision.

Examples

input

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

output

```
0.1250000000
0.2916666667
0.2916666667
0.2916666667
```

input

```
7
1 2
1 3
2 4
2 5
3 6
3 7
```

output

```
0.0850694444
0.0664062500
0.0664062500
0.1955295139
0.1955295139
0.1955295139
0.1955295139
```

Note

In the first sample, the resulting vertex has label 1 if and only if for all three edges the label 1 survives, hence the probability is $$$1/2^3 = 1/8$$$. All other labels have equal probability due to symmetry, hence each of them has probability $$$(1 - 1/8) / 3 = 7/24$$$.

G. Balls and Pockets

time limit per test: 2 seconds

memory limit per test: 512 megabytes

input: standard input

output: standard output

There is a strip with an infinite number of cells. Cells are numbered starting with $$$0$$$. Initially the cell $$$i$$$ contains a ball with the number $$$i$$$.

There are $$$n$$$ pockets located at cells $$$a_1, \ldots, a_n$$$. Each cell contains at most one pocket.

Filtering is the following sequence of operations:

- All pockets at cells $$$a_1, \ldots, a_n$$$ open simultaneously, which makes balls currently located at those cells disappear. After the balls disappear, the pockets close again.
- For each cell $$$i$$$ from $$$0$$$ to $$$\infty$$$, if the cell $$$i$$$ contains a ball, we move that ball to the free cell $$$j$$$ with the lowest number. If there is no free cell $$$j < i$$$, the ball stays at the cell $$$i$$$.

Note that after each filtering operation each cell will still contain exactly one ball.

For example, let the cells $$$1$$$, $$$3$$$ and $$$4$$$ contain pockets. The initial configuration of balls is shown below (underscores display the cells with pockets):

$$0\ \underline{1}\ 2\ \underline{3}\ \underline{4}\ 5\ 6\ 7\ 8\ 9\ \ldots$$

After opening and closing the pockets, balls 1, 3 and 4 disappear:

0 _ 2 _ _ 5 6 7 8 9 ...

After moving all the balls to the left, the configuration looks like this:

$$0\ \underline{2}\ 5\ \underline{6}\ \underline{7}\ 8\ 9\ 10\ 11\ 12\ \ldots$$

Another filtering repetition results in the following:

$$0\ \underline{5}\ 8\ \underline{9}\ \underline{10}\ 11\ 12\ 13\ 14\ 15\ \ldots$$

You have to answer $$$m$$$ questions. The $$$i$$$-th of these questions is "what is the number of the ball located at the cell $$$x_i$$$ after $$$k_i$$$ repetitions of the filtering operation?"

Input

The first line contains two integers $$$n$$$ and $$$m$$$ — the number of pockets and questions respectively ($$$1 \leq n, m \leq 10^5$$$).

The following line contains $$$n$$$ integers $$$a_1, \ldots, a_n$$$ — the numbers of cells containing pockets ($$$0 \leq a_1 < \ldots < a_n \leq 10^9$$$).

The following $$$m$$$ lines describe questions. The $$$i$$$-th of these lines contains two integers $$$x_i$$$ and $$$k_i$$$ ($$$0 \leq x_i, k_i \leq 10^9$$$).

Output

Print $$$m$$$ numbers — answers to the questions, in the same order as given in the input.

Example

input

```
3 15
1 3 4
0 0
1 0
2 0
3 0
4 0
0 1
1 1
2 1
3 1
4 1
0 2
1 2
2 2
3 2
4 2
```

output

```
0
1
2
3
4
0
2
5
6
7
0
5
8
9
10
```

H. Sophisticated Device

time limit per test: 2 seconds

memory limit per test: 512 megabytes

input: standard input

output: standard output

You are given integers $$$d$$$ and $$$p$$$, $$$p$$$ is prime.

Also you have a mysterious device. It has memory cells, each contains an integer between $$$0$$$ and $$$p-1$$$. Also two instructions are supported, addition and raising to the $$$d$$$-th power. $$$\textbf{Both are modulo}$$$ $$$p$$$.

The memory cells are numbered $$$1, 2, \dots, 5000$$$. Initially cells $$$1$$$ and $$$2$$$ contain integers $$$x$$$ and $$$y$$$, respectively ($$$0 \leqslant x, y \leq p - 1$$$). All other cells contain $$$\textbf{1}$$$s.

You can not directly access values in cells, and you $$$\textbf{don't know}$$$ values of $$$x$$$ and $$$y$$$ (but you know they are written in first two cells). You mission, should you choose to accept it, is to write a program using the available instructions to obtain the product $$$xy$$$ modulo $$$p$$$ in one of the cells. You program should work for all possible $$$x$$$ and $$$y$$$.

Addition instruction evaluates sum of values in two cells and writes it to third cell. This instruction is encoded by a string "`+ e1 e2 to`", which writes sum of values in cells `e1` and `e2` into cell `to`. Any values of `e1`, `e2`, `to` can coincide.

Second instruction writes the $$$d$$$-th power of a value in some cell to the target cell. This instruction is encoded by a string "`^ e to`". Values `e` and `to` can coincide, in this case value in the cell will be overwritten.

Last instruction is special, this is the return instruction, and it is encoded by a string "`f target`". This means you obtained values $$$xy \bmod p$$$ in the cell `target`. No instructions should be called after this instruction.

Provide a program that obtains $$$xy \bmod p$$$ and uses no more than $$$5000$$$ instructions (including the return instruction).

It is guaranteed that, under given constrains, a solution exists.

Input

The first line contains space-separated integers $$$d$$$ and $$$p$$$ ($$$2 \leqslant d \leqslant 10$$$, $$$d < p$$$, $$$3 \leqslant p \leqslant 10^9 + 9$$$, $$$p$$$ is prime).

Output

Output instructions, one instruction per line in the above format. There should be no more than $$$5000$$$ lines, and the last line should be the return instruction.

Note

This problem has **no sample tests**. A sample output is shown below. Note that this output is not supposed to be a solution to any testcase, and is there purely to illustrate the output format.

$$$\texttt{+ 1 1 3}\\ \texttt{^ 3 3}\\ \texttt{+ 3 2 2}\\ \texttt{+ 3 2 3}\\ \texttt{^ 3 1}\\ \texttt{f 1}$$$

Here's a step-by-step runtime illustration:

$$$\begin{array}{|c|c|c|c|} \hline \texttt{} & \text{cell 1} & \text{cell 2} & \text{cell 3} \\

\hline \text{initially} & x & y & 1 \\ \hline \texttt{+ 1 1 3} & x & y & 2x \\ \hline

\texttt{^ 3 3} & x & y & (2x)^d \\ \hline

\texttt{+ 3 2 2} & x & y + (2x)^d & (2x)^d \\ \hline

\texttt{+ 3 2 3} & x & y + (2x)^d & y + 2\cdot(2x)^d \\ \hline

\texttt{^ 3 1} & (y + 2\cdot(2x)^d)^d & y + (2x)^d & y + 2\cdot(2x)^d \\ \hline

\end{array}$$$

Suppose that $$$d = 2$$$ and $$$p = 3$$$. Since for $$$x = 0$$$ and $$$y = 1$$$ the returned result is $$$1 \neq 0 \cdot 1 \bmod 3$$$, this program would be judged as incorrect.