## IndiaHacks 2nd Elimination 2017 (unofficial, unrated mirror, ICPC rules)

# A. Binary Blocks

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

You are given an image, that can be represented with a 2-d $n$ by $m$ grid of pixels. Each pixel of the image is either on or off, denoted by the characters "0" or "1", respectively. You would like to compress this image. You want to choose an integer $k > 1$ and split the image into $k$ by $k$ blocks. If $n$ and $m$ are not divisible by $k$, the image is padded with only zeros on the right and bottom so that they are divisible by $k$. Each pixel in each individual block must have the same value. The given image may not be compressible in its current state. Find the minimum number of pixels you need to toggle (after padding) in order for the image to be compressible for some $k$. More specifically, the steps are to first choose $k$, then the image is padded with zeros, then, we can toggle the pixels so it is compressible for this $k$. The image must be compressible in that state.

### Input

The first line of input will contain two integers $n$, $m$ ($2 \le n, m \le 2\,500$), the dimensions of the image.

The next $n$ lines of input will contain a binary string with exactly $m$ characters, representing the image.

### Output

Print a single integer, the minimum number of pixels needed to toggle to make the image compressible.

**Example**

| input |
| --- |
| 3 5<br>00100<br>10110<br>11001 |
| **output** |
| 5 |

### Note

We first choose $k = 2$.

The image is padded as follows:

```
001000
101100
110010
000000
```

We can toggle the image to look as follows:

```
001100
001100
000000
000000
```

We can see that this image is compressible for $k = 2$.

# B. Diverging Directions

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a directed weighted graph with $n$ nodes and $2n - 2$ edges. The nodes are labeled from $1$ to $n$, while the edges are labeled from $1$ to $2n - 2$. The graph's edges can be split into two parts.

- The first $n - 1$ edges will form a rooted spanning tree, with node $1$ as the root. All these edges will point away from the root.
- The last $n - 1$ edges will be from node $i$ to node $1$, for all $2 \leq i \leq n$.

You are given $q$ queries. There are two types of queries

- $1\ i\ w$: Change the weight of the $i$-th edge to $w$
- $2\ u\ v$: Print the length of the shortest path between nodes $u$ to $v$

Given these queries, print the shortest path lengths.

## Input

The first line of input will contain two integers $n$, $q$ ($2 \leq n$, $q \leq 200\,000$), the number of nodes, and the number of queries, respectively.

The next $2n - 2$ integers will contain 3 integers $a_i$, $b_i$, $c_i$, denoting a directed edge from node $a_i$ to node $b_i$ with weight $c_i$.

The first $n - 1$ of these lines will describe a rooted spanning tree pointing away from node $1$, while the last $n - 1$ of these lines will have $b_i = 1$.

More specifically,

- The edges $(a_1, b_1)$, $(a_2, b_2)$, ... $(a_{n-1}, b_{n-1})$ will describe a rooted spanning tree pointing away from node $1$.
- $b_j = 1$ for $n \leq j \leq 2n - 2$.
- $a_n, a_{n+1}, \ldots, a_{2n-2}$ will be distinct and between $2$ and $n$.

The next $q$ lines will contain 3 integers, describing a query in the format described in the statement.

All edge weights will be between $1$ and $10^6$.

## Output

For each type 2 query, print the length of the shortest path in its own line.

**Example**

| input |
| --- |
| 5 9<br>1 3 1<br>3 2 2<br>1 4 3<br>3 5 4<br>5 1 5<br>3 1 6<br>2 1 7<br>4 1 8<br>2 1 1<br>2 1 3<br>2 3 5<br>2 5 2<br>1 1 100<br>2 1 3<br>1 8 30<br>2 4 2<br>2 2 4 |

| output |
| --- |
| 0<br>1<br>4<br>8<br>100<br>132<br>10 |

# C. Future Failure

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alice and Bob are playing a game with a string of characters, with Alice going first. The string consists $n$ characters, each of which is one of the first $k$ letters of the alphabet. On a player's turn, they can either arbitrarily permute the characters in the words, or delete exactly one character in the word (if there is at least one character). In addition, their resulting word cannot have appeared before throughout the entire game. The player unable to make a valid move loses the game.

Given $n$, $k$, $p$, find the number of words with exactly $n$ characters consisting of the first $k$ letters of the alphabet such that Alice will win if both Alice and Bob play optimally. Return this number modulo the prime number $p$.

## Input

The first line of input will contain three integers $n$, $k$, $p$ ($1 \le n \le 250\,000$, $1 \le k \le 26$, $10^8 \le p \le 10^9 + 100$, $p$ will be prime).

## Output

Print a single integer, the number of winning words for Alice, modulo $p$.

**Example**

| input |
| --- |
| 4 2 100000007 |
| **output** |
| 14 |

## Note

There are 14 strings that that Alice can win with. For example, some strings are "bbaa" and "baaa". Alice will lose on strings like "aaaa" or "bbbb".

# D. Airplane Arrangements

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

There is an airplane which has $n$ rows from front to back. There will be $m$ people boarding this airplane.

This airplane has an entrance at the very front and very back of the plane.

Each person has some assigned seat. It is possible for multiple people to have the same assigned seat. The people will then board the plane one by one starting with person $1$. Each person can independently choose either the front entrance or back entrance to enter the plane.

When a person walks into the plane, they will walk directly to their assigned seat. Then, while the seat they're looking at is occupied, they will keep moving one space in the same direction. A passenger will get angry if they reach the end of the row without finding their assigned seat.

Find the number of ways to assign tickets to the passengers and board the plane without anyone getting angry. Two ways are different if there exists a passenger who chose a different entrance in both ways, or the assigned seat is different. Print this count modulo $10^9 + 7$.

## Input

The first line of input will contain two integers $n$, $m$ ($1 \le m \le n \le 1\,000\,000$), the number of seats, and the number of passengers, respectively.

## Output

Print a single number, the number of ways, modulo $10^9 + 7$.

**Example**

| input |
| --- |
| 3 3 |
| **output** |
| 128 |

## Note

Here, we will denote a passenger by which seat they were assigned, and which side they came from (either "F" or "B" for front or back, respectively).

For example, one valid way is 3B, 3B, 3B (i.e. all passengers were assigned seat 3 and came from the back entrance). Another valid way would be 2F, 1B, 3F.

One invalid way would be 2B, 2B, 2B, since the third passenger would get to the front without finding a seat.

# E. Convex Countour

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

You are given an strictly convex polygon with $n$ vertices. It is guaranteed that no three points are collinear. You would like to take a maximum non intersecting path on the polygon vertices that visits each point at most once.

More specifically your path can be represented as some sequence of distinct polygon vertices. Your path is the straight line segments between adjacent vertices in order. These segments are not allowed to touch or intersect each other except at the vertices in your sequence.

Given the polygon, print the maximum length non-intersecting path that visits each point at most once.

## Input

The first line of input will contain a single integer $n$ ($2 \leq n \leq 2\,500$), the number of points.

The next $n$ lines will contain two integers $x_i$, $y_i$ ($|x_i|$, $|y_i| \leq 10^9$), denoting the coordinates of the $i$-th vertex.

It is guaranteed that these points are listed in clockwise order.

## Output

Print a single floating point number, representing the longest non-intersecting path that visits the vertices at most once.

Your answer will be accepted if it has absolute or relative error at most $10^{-9}$. More specifically, if your answer is $a$ and the jury answer is $b$, your answer will be accepted if $\frac{|a-b|}{\max(1,b)} \leq 10^{-9}$.

**Example**

| input |
| --- |
| 4<br>0 0<br>0 1<br>1 1<br>1 0 |
| **output** |
| 3.4142135624 |

## Note

One optimal path is to visit points 0,1,3,2 in order.

# F. Expected Earnings

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

You are playing a game with a bag of red and black balls. Initially, you are told that the bag has $n$ balls total. In addition, you are also told that the bag has probability $p_i / 10^6$ of containing exactly $i$ red balls.

You now would like to buy balls from this bag. You really like the color red, so red balls are worth a unit of $1$, while black balls are worth nothing. To buy a ball, if there are still balls in the bag, you pay a cost $c$ with $0 \le c \le 1$, and draw a ball randomly from the bag. You can choose to stop buying at any point (and you can even choose to not buy anything at all).

Given that you buy optimally to maximize the expected profit (i.e. # red balls - cost needed to obtain them), print the maximum expected profit.

## Input

The first line of input will contain two integers $n$, $X$ ($1 \le n \le 10\,000$, $0 \le X \le 10^6$).

The next line of input will contain $n + 1$ integers $p_0$, $p_1$, ... $p_n$ ($0 \le p_i \le 10^6$, $\sum_{i=0}^{n} p_i = 10^6$)

The value of $c$ can be computed as $\frac{X}{10^6}$.

## Output

Print a single floating point number representing the optimal expected value.

Your answer will be accepted if it has absolute or relative error at most $10^{-9}$. More specifically, if your answer is $a$ and the jury answer is $b$, your answer will be accepted if $\frac{|a-b|}{\max(1,b)} \le 10^{-9}$.

**Example**

| input |
| --- |
| 3 200000<br>250000 250000 250000 250000 |
| **output** |
| 0.9000000000 |

## Note

Here, there is equal probability for the bag to contain 0,1,2,3 red balls. Also, it costs 0.2 to draw a ball from the bag.