A. Game Shopping

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

Maxim wants to buy some games at the local game shop. There are n games in the shop, the i-th game costs c_i.

Maxim has a wallet which can be represented as an array of integers. His wallet contains m bills, the j-th bill has value a_j.

Games in the shop are ordered from left to right, Maxim tries to buy *every* game in that order.

When Maxim stands at the position i in the shop, he takes the first bill from his wallet (if his wallet is empty then he proceeds to the next position immediately) and tries to buy the i-th game using this bill. After Maxim tried to buy the n-th game, he leaves the shop.

Maxim buys the i-th game if and only if the value of the first bill (which he takes) from his wallet is greater or equal to the cost of the i-th game. If he successfully buys the i-th game, the first bill from his wallet disappears and the next bill becomes first. Otherwise Maxim leaves the first bill in his wallet **(this bill still remains the first one)** and proceeds to the next game.

For example, for array $c = [2, 4, 5, 2, 4]$ and array $a = [5, 3, 4, 6]$ the following process takes place: Maxim buys the first game using the first bill (its value is 5), the bill disappears, after that the second bill (with value 3) becomes the first one in Maxim's wallet, then Maxim doesn't buy the second game because $c_2 > a_2$, the same with the third game, then he buys the fourth game using the bill of value a_2 (the third bill becomes the first one in Maxim's wallet) and buys the fifth game using the bill of value a_3.

Your task is to get the number of games Maxim will buy.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 1000$) — the number of games and the number of bills in Maxim's wallet.

The second line of the input contains n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le 1000$), where c_i is the cost of the i-th game.

The third line of the input contains m integers $a_1, a_2, \ldots, a_m$ ($1 \le a_j \le 1000$), where a_j is the value of the j-th bill from the Maxim's wallet.

Output

Print a single integer — the number of games Maxim will buy.

Examples

input
5 4 2 4 5 2 4 5 3 4 6
output
3

input
5 2 20 40 50 20 40 19 20
output
0

input
6 4 4 8 15 16 23 42 1000 1000 1000 1000
output
4

Note

The first example is described in the problem statement.

In the second example Maxim cannot buy any game because the value of the first bill in his wallet is smaller than the cost of any game in the shop.

In the third example the values of the bills in Maxim's wallet are large enough to buy any game he encounter until he runs out of bills in his wallet.

B. Minimum Ternary String

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

You are given a ternary string (it is a string which consists only of characters '0', '1' and '2').

You can swap any two adjacent (consecutive) characters '0' and '1' (i.e. replace "01" with "10" or vice versa) or any two adjacent (consecutive) characters '1' and '2' (i.e. replace "12" with "21" or vice versa).

For example, for string "010210" we can perform the following moves:

- "01̲0210" → "10̲0210";
- "0̲10210" → "00̲1210";
- "0102̲10" → "0102̲01";
- "01021̲0" → "01012̲0".

Note than you cannot swap "02̲" → "20̲" and vice versa. You cannot perform any other operations with the given string excluding described above.

You task is to obtain the minimum possible (lexicographically) string by using these swaps arbitrary number of times (*possibly, zero*).

String a is lexicographically less than string b (if strings a and b have the same length) if there exists some position i ($1 \le i \le |a|$, where $|s|$ is the length of the string s) such that for every $j < i$ holds $a_j = b_j$, and $a_i < b_i$.

Input

The first line of the input contains the string s consisting only of characters '0', '1' and '2', its length is between 1 and 10^5 (inclusive).

Output

Print a single string — the minimum possible (lexicographically) string you can obtain by using the swaps described above arbitrary number of times (*possibly, zero*).

Examples

input
100210
output
001120

input
11222121
output
11112222

input
20
output
20

C. Annoying Present

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

Alice got an array of length n as a birthday present once again! This is the third year in a row!

And what is more disappointing, it is overwhelmingly boring, filled entirely with zeros. Bob decided to apply some changes to the array to cheer up Alice.

Bob has chosen m changes of the following form. For some integer numbers x and d, he chooses an arbitrary position i ($1 \le i \le n$) and for every $j \in [1, n]$ adds $x + d \cdot dist(i, j)$ to the value of the j-th cell. $dist(i, j)$ is the distance between positions i and j (i.e. $dist(i, j) = |i - j|$, where $|x|$ is an absolute value of x).

For example, if Alice currently has an array $[2, 1, 2, 2]$ and Bob chooses position 3 for $x = -1$ and $d = 2$ then the array will become

$[2-1+2\cdot 2,\ 1-1+2\cdot 1,\ 2-1+2\cdot 0,\ 2-1+2\cdot 1] = [5,2,1,3]$. Note that Bob can't choose position i outside of the array (that is, smaller than 1 or greater than n).

Alice will be the happiest when the elements of the array are as big as possible. Bob claimed that the arithmetic mean value of the elements will work fine as a metric.

What is the maximum arithmetic mean value Bob can achieve?

Input
The first line contains two integers n and m ($1 \le n, m \le 10^5$) — the number of elements of the array and the number of changes.

Each of the next m lines contains two integers x_i and d_i ($-10^3 \le x_i, d_i \le 10^3$) — the parameters for the i-th change.

Output
Print the maximal average arithmetic mean of the elements Bob can achieve.

Your answer is considered correct if its absolute or relative error doesn't exceed 10^{-6}.

Examples

input
2 3 -1 3 0 0 -1 -4
output
-2.500000000000000

input
3 2 0 2 5 0
output
7.000000000000000

D. Relatively Prime Graph

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

Let's call an undirected graph $G = (V, E)$ *relatively prime* if and only if for each edge $(v, u) \in E$ $GCD(v, u) = 1$ (the greatest common divisor of v and u is 1). If there is no edge between some pair of vertices v and u then the value of $GCD(v, u)$ doesn't matter. The vertices are numbered from 1 to $|V|$.

Construct a *relatively prime* graph with n vertices and m edges such that it is connected and it contains neither self-loops nor multiple edges.

If there exists no valid graph with the given number of vertices and edges then output `"Impossible"`.

If there are multiple answers then print any of them.

Input
The only line contains two integers n and m ($1 \le n, m \le 10^5$) — the number of vertices and the number of edges.

Output
If there exists no valid graph with the given number of vertices and edges then output `"Impossible"`.

Otherwise print the answer in the following format:

The first line should contain the word `"Possible"`.

The i-th of the next m lines should contain the i-th edge (v_i, u_i) of the resulting graph ($1 \le v_i, u_i \le n, v_i \ne u_i$). For each pair (v, u) there can be no more pairs (v, u) or (u, v). The vertices are numbered from 1 to n.

If there are multiple answers then print any of them.

Examples

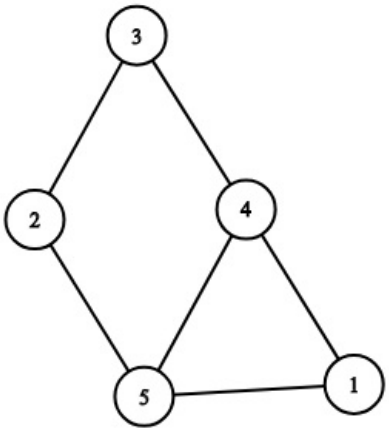
input
5 6
output
Possible 2 5 3 2

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

input
6 12
output
Impossible

Note

Here is the representation of the graph from the first example:



E. Intercity Travelling

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Leha is planning his journey from Moscow to Saratov. He hates trains, so he has decided to get from one city to another by car.

The path from Moscow to Saratov can be represented as a straight line (well, it's not that straight in reality, but in this problem we will consider it to be straight), and the distance between Moscow and Saratov is n km. Let's say that Moscow is situated at the point with coordinate 0 km, and Saratov — at coordinate n km.

Driving for a long time may be really difficult. Formally, if Leha has already covered i kilometers since he stopped to have a rest, he considers the *difficulty of covering* $(i+1)$-th kilometer as a_{i+1}. It is guaranteed that for every $i \in [1, n-1]$ $a_i \le a_{i+1}$. The difficulty of the journey is denoted as the sum of difficulties of each kilometer in the journey.

Fortunately, there may be some rest sites between Moscow and Saratov. Every integer point from 1 to $n-1$ may contain a rest site. When Leha enters a rest site, he may have a rest, and the next kilometer will have difficulty a_1, the kilometer after it — difficulty a_2, and so on.

For example, if $n = 5$ and there is a rest site in coordinate 2, the difficulty of journey will be $2a_1 + 2a_2 + a_3$: the first kilometer will have difficulty a_1, the second one — a_2, then Leha will have a rest, and the third kilometer will have difficulty a_1, the fourth — a_2, and the last one — a_3. Another example: if $n = 7$ and there are rest sites in coordinates 1 and 5, the difficulty of Leha's journey is $3a_1 + 2a_2 + a_3 + a_4$.

Leha doesn't know which integer points contain rest sites. So he has to consider every possible situation. Obviously, there are 2^{n-1} different distributions of rest sites (two distributions are different if there exists some point x such that it contains a rest site in exactly one of these distributions). Leha considers all these distributions to be equiprobable. He wants to calculate p — the expected value of difficulty of his journey.

Obviously, $p \cdot 2^{n-1}$ is an integer number. You have to calculate it modulo 998244353.

Input

The first line contains one number n ($1 \le n \le 10^6$) — the distance from Moscow to Saratov.

The second line contains n integer numbers a_1, a_2, ..., a_n ($1 \le a_1 \le a_2 \le \dots \le a_n \le 10^6$), where a_i is the difficulty of i-th kilometer after Leha has rested.

Output

Print one number — $p \cdot 2^{n-1}$, taken modulo 998244353.

Examples

input
2 1 2

output
5

input
4 1 3 3 7
output
60

F. Dominant Indices

time limit per test: 4.5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given a rooted undirected tree consisting of n vertices. Vertex 1 is the root.

Let's denote a *depth array* of vertex x as an infinite sequence $[d_{x,0}, d_{x,1}, d_{x,2}, \ldots]$, where $d_{x,i}$ is the number of vertices y such that both conditions hold:

- x is an ancestor of y;
- the simple path from x to y traverses exactly i edges.

The *dominant index* of a *depth array* of vertex x (or, shortly, the *dominant index* of vertex x) is an index j such that:

- for every $k < j$, $d_{x,k} < d_{x,j}$;
- for every $k > j$, $d_{x,k} \le d_{x,j}$.

For every vertex in the tree calculate its *dominant index*.

Input

The first line contains one integer n ($1 \le n \le 10^6$) — the number of vertices in a tree.

Then $n - 1$ lines follow, each containing two integers x and y ($1 \le x, y \le n$, $x \ne y$). This line denotes an edge of the tree.

It is guaranteed that these edges form a tree.

Output

Output n numbers. i-th number should be equal to the *dominant index* of vertex i.

Examples

input
4 1 2 2 3 3 4
output
0 0 0 0

input
4 1 2 1 3 1 4
output
1 0 0 0

input
4 1 2 2 3 2 4
output
2 1 0

0

G. Allowed Letters

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

Polycarp has just launched his new startup idea. The niche is pretty free and the key vector of development sounds really promising, so he easily found himself some investors ready to sponsor the company. However, he is yet to name the startup!

Actually, Polycarp has already came up with the name but some improvement to it will never hurt. So now he wants to swap letters at some positions in it to obtain the better name. It isn't necessary for letters to be adjacent.

In addition, each of the investors has chosen some index in the name and selected a set of letters that can go there. Indices chosen by different investors are pairwise distinct. If some indices aren't chosen by any investor then any letter can go there.

Finally, Polycarp is sure that the smallest lexicographically name is the best. (Like why do you think Google decided to become Alphabet?)

More formally, you are given a string consisting of lowercase Latin letters from "a" to "f". You can swap letters at any positions arbitrary number of times (zero swaps is also possible).

What is the smallest lexicographically name you can obtain such that the letter at every position is among the allowed letters?

If Polycarp can't produce any valid name then print "Impossible".

Input

The first line is the string s ($1 \le |s| \le 10^5$) — the name Polycarp has came up with. The string consists only of lowercase Latin letters from "a" to "f".

The second line contains a single integer m ($0 \le m \le |s|$) — the number of investors.

The i-th of the next m lines contain an integer number pos_i and a non-empty string of allowed characters for pos_i ($1 \le pos_i \le |s|$). Each string contains pairwise distinct letters from "a" to "f". $pos_1, pos_2, \ldots, pos_m$ are pairwise distinct. If any position of the string doesn't appear in the investors demands then any letter can go in this position.

Output

If Polycarp can't produce any valid name then print "Impossible".

Otherwise print the smallest lexicographically name Polycarp can obtain by swapping letters in string s such that the letter at every position is among the allowed ones.

Examples

input
bedefead 5 2 e 1 dc 5 b 7 ef 6 ef
output
deadbeef

input
abacaba 0
output
aaaabbc

input
fc 2 1 cfab 2 f
output
cf