Educational Codeforces Round 25

A. Binary Protocol

1 second, 256 megabytes

Polycarp has just invented a new binary protocol for data transmission. He is encoding positive integer decimal number to binary string using following algorithm:

- Each digit is represented with number of '1' characters equal to the value of that digit (for 0 it is zero ones).
- Digits are written one by one in order corresponding to number and separated by single '0' character.

Though Polycarp learnt how to encode the numbers, he has no idea how to decode them back. Help him calculate the decoded number.

Input

The first line contains one integer number n $(1 \le n \le 89)$ — length of the string s.

The second line contains string s — sequence of '0' and '1' characters, number in its encoded format. It is guaranteed that the number corresponding to the string is positive and doesn't exceed 10^9. The string always starts with '1'.

Output

Print the decoded number.

input

```
3
111
```

output

```
3
```

input

```
9
110011101
```

output

```
2031
```

B. Five-In-a-Row

1 second, 256 megabytes

Alice and Bob play 5-in-a-row game. They have a playing field of size 10×10. In turns they put either crosses or noughts, one at a time. Alice puts crosses and Bob puts noughts.

In current match they have made some turns and now it's Alice's turn. She wonders if she can put cross in such empty cell that she wins immediately.

Alice wins if some crosses in the field form line of length **not smaller than** 5. This line can be horizontal, vertical and diagonal.

Input

You are given matrix 10×10 (10 lines of 10 characters each) with capital Latin letters 'X' being a cross, letters 'O' being a nought and '.' being an empty cell. The number of 'X' cells is equal to the number of 'O' cells and there is at least one of each type. There is at least one empty cell.

It is guaranteed that in the current arrangement nobody has still won.

Output

Print 'YES' if it's possible for Alice to win in one turn by putting cross in some empty cell. Otherwise print 'NO'.

input

```
XX.XX.....
.....OOOO.
..........
..........
..........
..........
..........
..........
..........
..........
```

output

```
YES
```

input

```
XXOXX.....
OO.O......
..........
..........
..........
..........
..........
..........
..........
..........
```

output

```
NO
```

C. Multi-judge Solving

1 second, 256 megabytes

Makes solves problems on Decoforces and lots of other different online judges. Each problem is denoted by its difficulty — a positive integer number. Difficulties are measured the same across all the judges (the problem with difficulty d on Decoforces is as hard as the problem with difficulty d on any other judge).

Makes has chosen n problems to solve on Decoforces with difficulties $a_1, a_2, ..., a_n$. He can solve these problems in arbitrary order. Though he can solve problem i with difficulty a_i only if he had already solved some problem with difficulty $d \geq \frac{a_i}{2}$ (no matter on what online judge was it).

Before starting this chosen list of problems, Makes has already solved problems with maximum difficulty k.

With given conditions it's easy to see that Makes sometimes can't solve all the chosen problems, no matter what order he chooses. So he wants to solve some problems on other judges to finish solving problems from his list.

For every positive integer y there exist some problem with difficulty y on at least one judge besides Decoforces.

Makes can solve problems on any judge at any time, it isn't necessary to do problems from the chosen list one right after another.

Makes doesn't have too much free time, so he asked you to calculate the minimum number of problems he should solve on other judges in order to solve all the chosen problems from Decoforces.

Input

The first line contains two integer numbers n, k $(1 \leq n \leq 10^3, 1 \leq k \leq 10^9)$.

The second line contains n space-separated integer numbers $a_1, a_2, ..., a_n$ $(1 \leq a_i \leq 10^9)$.

Output

Print minimum number of problems Makes should solve on other judges in order to solve all chosen problems on Decoforces.

input

```
3 3
2 1 9
```

output

```
1
```

input

```
4 20
10 3 6 3
```

output

```
0
```

In the first example Makes at first solves problems 1 and 2. Then in order to solve the problem with difficulty 9, he should solve problem with difficulty no less than 5. The only available are difficulties 5 and 6 on some other judge. Solving any of these will give Makes opportunity to solve problem 3.

In the second example he can solve every problem right from the start.

D. Suitable Replacement

1 second, 256 megabytes

You are given two strings s and t consisting of small Latin letters, string s can also contain '?' characters.

Suitability of string s is calculated by following metric:

Any two letters can be swapped positions, these operations can be performed arbitrary number of times over any pair of positions. Among all resulting strings s, you choose the one with the largest number of **non-intersecting** occurrences of string t. *Suitability* is this number of occurrences.

You should replace all '?' characters with small Latin letters in such a way that the *suitability* of string s is maximal.

Input

The first line contains string s ($1 \le |s| \le 10^6$).

The second line contains string t ($1 \le |t| \le 10^6$).

Output

Print string s with '?' replaced with small Latin letters in such a way that *suitability* of that string is maximal.

If there are multiple strings with maximal *suitability* then print any of them.

input
```
?aa?
ab
```
output
```
baab
```

input
```
??b?
za
```
output
```
azbz
```

input
```
abcd
abacaba
```
output
```
abcd
```

In the first example string "baab" can be transformed to "abab" with swaps, this one has *suitability* of 2. That means that string "baab" also has *suitability* of 2.

In the second example maximal *suitability* you can achieve is 1 and there are several dozens of such strings, "azbz" is just one of them.

In the third example there are no '?' characters and the *suitability* of the string is 0.

E. Minimal Labels

1 second, 256 megabytes

You are given a directed acyclic graph with n vertices and m edges. There are no self-loops or multiple edges between any pair of vertices. Graph can be disconnected.

You should assign labels to all vertices in such a way that:

- Labels form a valid permutation of length n — an integer sequence such that each integer from 1 to n appears exactly once in it.
- If there exists an edge from vertex v to vertex u then $label_v$ should be smaller than $label_u$.
- Permutation should be lexicographically smallest among all suitable.

Find such sequence of labels to satisfy all the conditions.

Input

The first line contains two integer numbers n, m ($2 \le n \le 10^5, 1 \le m \le 10^5$).

Next m lines contain two integer numbers v and u ($1 \le v, u \le n, v \ne u$) — edges of the graph. Edges are directed, graph doesn't contain loops or multiple edges.

Output

Print n numbers — lexicographically smallest correct permutation of labels of vertices.

input

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

output

```
1 3 2
```

input

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

output

```
4 1 2 3
```

input

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

output

```
3 1 2 4 5
```

F. String Compression

2 seconds, 512 megabytes

Ivan wants to write a letter to his friend. The letter is a string s consisting of lowercase Latin letters.

Unfortunately, when Ivan started writing the letter, he realised that it is very long and writing the whole letter may take extremely long time. So he wants to write the *compressed version* of string s instead of the string itself.

The *compressed version* of string s is a sequence of strings $c_1, s_1, c_2, s_2, ..., c_k, s_k$, where c_i is the decimal representation of number a_i (without any leading zeroes) and s_i is some string consisting of lowercase Latin letters. If Ivan writes string s_1 exactly a_1 times, then string s_2 exactly a_2 times, and so on, the result will be string s.

The length of a *compressed version* is $|c_1| + |s_1| + |c_2| + |s_2|... |c_k| + |s_k|$. Among all *compressed versions* Ivan wants to choose a version such that its length is minimum possible. Help Ivan to determine minimum possible length.

Input

The only line of input contains one string s consisting of lowercase Latin letters ($1 \le |s| \le 8000$).

Output

Output one integer number — the minimum possible length of a *compressed version* of s.

input

```
aaaaaaaaaa
```

output

```
3
```

input

```
abcab
```

output

```
6
```

input

```
cczabababab
```

output

```
7
```

In the first example Ivan will choose this compressed version: c_1 is `10`, s_1 is `a`.

In the second example Ivan will choose this compressed version: c_1 is `1`, s_1 is `abcab`.

In the third example Ivan will choose this compressed version: c_1 is `2`, s_1 is `c`, c_2 is `1`, s_2 is `z`, c_3 is `4`, s_3 is `ab`.

G. Tree Queries

3 seconds, 256 megabytes

You are given a tree consisting of n vertices (numbered from 1 to n). Initially all vertices are white. You have to process q queries of two different types:

1. $1\ x$ — change the color of vertex x to black. It is guaranteed that the first query will be of this type.
2. $2\ x$ — for the vertex x, find the minimum index y such that the vertex with index y belongs to the simple path from x to some black vertex (a simple path never visits any vertex more than once).

For each query of type 2 print the answer to it.

Note that the queries are given in modified way.

Input

The first line contains two numbers n and q ($3 \le n, q \le 10^6$).

Then n - 1 lines follow, each line containing two numbers x_i and y_i ($1 \le x_i < y_i \le n$) and representing the edge between vertices x_i and y_i.

It is guaranteed that these edges form a tree.

Then q lines follow. Each line contains two integers t_i and z_i, where t_i is the type of ith query, and z_i can be used to restore x_i for this query in this way: you have to keep track of the answer to the last query of type 2 (let's call this answer $last$, and initially $last = 0$); then $x_i = (z_i + last) \bmod n + 1$.

It is guaranteed that the first query is of type 1, and there is at least one query of type 2.

Output

For each query of type 2 output the answer to it.

input

```
4 6
1 2
2 3
3 4
1 2
1 2
2 2
1 3
2 2
2 2
```

output

```
3
2
1
```