Codeforces Round #112 (Div. 2)

A. Supercentral Point

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

One day Vasya painted a Cartesian coordinate system on a piece of paper and marked some set of points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. Let's define neighbors for some fixed point from the given set (x, y):

- point (x', y') is (x, y)'s right neighbor, if $x' > x$ and $y' = y$
- point (x', y') is (x, y)'s left neighbor, if $x' < x$ and $y' = y$
- point (x', y') is (x, y)'s lower neighbor, if $x' = x$ and $y' < y$
- point (x', y') is (x, y)'s upper neighbor, if $x' = x$ and $y' > y$

We'll consider point (x, y) from the given set supercentral, if it has at least one upper, at least one lower, at least one left and at least one right neighbor among this set's points.

Vasya marked quite many points on the paper. Analyzing the picture manually is rather a challenge, so Vasya asked you to help him. Your task is to find the number of supercentral points in the given set.

Input

The first input line contains the only integer n ($1 \le n \le 200$) — the number of points in the given set. Next n lines contain the coordinates of the points written as "x y" (without the quotes) ($|x|, |y| \le 1000$), all coordinates are integers. The numbers in the line are separated by exactly one space. It is guaranteed that all points are different.

Output

Print the only number — the number of supercentral points of the given set.

Examples

input
8 1 1 4 2 3 1 1 2 0 2 0 1 1 0 1 3
output
2

input
5 0 0 0 1 1 0 0 -1 -1 0
output
1

Note

In the first sample the supercentral points are only points $(1, 1)$ and $(1, 2)$.

In the second sample there is one supercental point — point $(0, 0)$.

B. Burning Midnight Oil

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

One day a highly important task was commissioned to Vasya — writing a program in a night. The program consists of n lines of code. Vasya is already exhausted, so he works like that: first he writes v lines of code, drinks a cup of tea, then he writes as much as $\lfloor \frac{v}{k} \rfloor$ lines, drinks another cup of tea, then he writes $\lfloor \frac{v}{k^2} \rfloor$ lines and so on: $\lfloor \frac{v}{k^3} \rfloor$, $\lfloor \frac{v}{k^4} \rfloor$, $\lfloor \frac{v}{k^5} \rfloor$, ...

The expression $\lfloor \frac{a}{b} \rfloor$ is regarded as the integral part from dividing number a by number b.

The moment the current value $\lfloor \frac{v}{k^p} \rfloor$ equals 0, Vasya immediately falls asleep and he wakes up only in the morning, when the program should already be finished.

Vasya is wondering, what minimum allowable value v can take to let him write **not less** than n lines of code before he falls asleep.

Input

The input consists of two integers n and k, separated by spaces — the size of the program in lines and the productivity reduction coefficient, $1 \le n \le 10^9$, $2 \le k \le 10$.

Output

Print the only integer — the minimum value of v that lets Vasya write the program in one night.

Examples

input
7 2
output
4

input
59 9
output
54

Note

In the first sample the answer is $v = 4$. Vasya writes the code in the following portions: first 4 lines, then 2, then 1, and then Vasya falls asleep. Thus, he manages to write $4 + 2 + 1 = 7$ lines in a night and complete the task.

In the second sample the answer is $v = 54$. Vasya writes the code in the following portions: 54, 6. The total sum is $54 + 6 = 60$, that's even more than $n = 59$.

C. Another Problem on Strings

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

A string is *binary*, if it consists only of characters "0" and "1".

String V is a *substring* of string W if it has a non-zero length and can be read starting from some position in string W. For example, string "010" has six substrings: "0", "1", "0", "01", "10", "010". Two substrings are considered different if their positions of occurrence are different. So, if some string occurs multiple times, we should consider it the number of times it occurs.

You are given a binary string S. Your task is to find the number of its substrings, containing exactly K characters "1".

Input

The first line contains the single integer K ($0 \le K \le 10^6$). The second line contains a non-empty binary string S. The length of S does not exceed 10^6 characters.

Output

Print the single number — the number of substrings of the given string, containing exactly K characters "1".

Please do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Examples

input
1 1010
output
6

input
2 01010
output
4

input
100 01010
output
0

Note

In the first sample the sought substrings are: "1", "1", "10", "01", "10", "010".

In the second sample the sought substrings are: "101", "0101", "1010", "01010".

D. Beard Graph

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's define a non-oriented connected graph of n vertices and $n - 1$ edges as a *beard*, if all of its vertices except, perhaps, one, have the degree of 2 or 1 (that is, there exists no more than one vertex, whose degree is more than two). Let us remind you that the degree of a vertex is the number of edges that connect to it.

Let each edge be either black or white. Initially all edges are black.

You are given the description of the beard graph. Your task is to analyze requests of the following types:

- paint the edge number i black. The edge number i is the edge that has this number in the description. It is guaranteed that by the moment of this request the i-th edge is white
- paint the edge number i white. It is guaranteed that by the moment of this request the i-th edge is black
- find the length of the shortest path going **only along the black edges** between vertices a and b or indicate that no such path exists between them (a path's length is the number of edges in it)

The vertices are numbered with integers from 1 to n, and the edges are numbered with integers from 1 to $n - 1$.

Input

The first line of the input contains an integer n ($2 \le n \le 10^5$) — the number of vertices in the graph. Next $n - 1$ lines contain edges described as the numbers of vertices v_i, u_i ($1 \le v_i, u_i \le n$, $v_i \ne u_i$) connected by this edge. It is guaranteed that the given graph is connected and forms a beard graph, and has no self-loops or multiple edges.

The next line contains an integer m ($1 \le m \le 3 \cdot 10^5$) — the number of requests. Next m lines contain requests in the following form: first a line contains an integer $type$, which takes values from 1 to 3, and represents the request type.

If $type = 1$, then the current request is a request to paint the edge black. In this case, in addition to number $type$ the line should contain integer id ($1 \le id \le n - 1$), which represents the number of the edge to paint.

If $type = 2$, then the current request is a request to paint the edge white, its form is similar to the previous request.

If $type = 3$, then the current request is a request to find the distance. In this case, in addition to $type$, the line should contain two integers a, b ($1 \le a, b \le n$, a can be equal to b) — the numbers of vertices, the distance between which must be found.

The numbers in all lines are separated by exactly one space. The edges are numbered in the order in which they are given in the input.

Output

For each request to "find the distance between vertices a and b" print the result. If there is no path going only along the black edges between vertices a and b, then print "-1" (without the quotes). Print the results in the order of receiving the requests, separate the numbers with spaces or line breaks.

Examples

input
3 1 2 2 3 7 3 1 2 3 1 3 3 2 3 2 2 3 1 2 3 1 3 3 2 3

output
1 2 1 1 -1 -1

input
6 1 5 6 4 2 3

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

output

```
3
-1
3
2
```

Note

In the first sample vertices 1 and 2 are connected with edge number 1, and vertices 2 and 3 are connected with edge number 2. Before the repainting edge number 2 each vertex is reachable from each one along the black edges. Specifically, the shortest path between 1 and 3 goes along both edges.

If we paint edge number 2 white, vertex 3 will end up cut off from other vertices, that is, no path exists from it to any other vertex along the black edges.

E. Compatible Numbers

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Two integers x and y are *compatible*, if the result of their bitwise "AND" equals zero, that is, $a \& b = 0$. For example, numbers 90 (1011010_2) and 36 (100100_2) are compatible, as $1011010_2 \& 100100_2 = 0_2$, and numbers 3 (11_2) and 6 (110_2) are not compatible, as $11_2 \& 110_2 = 10_2$.

You are given an array of integers $a_1, a_2, \ldots, a_n$. Your task is to find the following for each array element: is this element compatible with some other element from the given array? If the answer to this question is positive, then you also should find any suitable element.

Input

The first line contains an integer n ($1 \le n \le 10^6$) — the number of elements in the given array. The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 4 \cdot 10^6$) — the elements of the given array. The numbers in the array can coincide.

Output

Print n integers ans_i. If a_i isn't compatible with any other element of the given array $a_1, a_2, \ldots, a_n$, then ans_i should be equal to -1. Otherwise ans_i is any such number, that $a_i \& ans_i = 0$, and also ans_i occurs in the array $a_1, a_2, \ldots, a_n$.

Examples

input
2 90 36
output
36 90

input
4 3 6 3 6
output
-1 -1 -1 -1

input
5 10 6 9 8 2
output
-1 8 2 2 8