A

Hongcow Builds A Nation

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Hongcow is ruler of the world. As ruler of the world, he wants to make it easier for people to travel by road within their own countries.

The world can be modeled as an undirected graph with n nodes and m edges. k of the nodes are home to the governments of the k countries that make up the world.

There is at most one edge connecting any two nodes and no edge connects a node to itself. Furthermore, for any two nodes corresponding to governments, **there is no path between those two nodes**. Any graph that satisfies all of these conditions is *stable*.

Hongcow wants to add as many edges as possible to the graph while keeping it stable. Determine the maximum number of edges Hongcow can add.

Input

The first line of input will contain three integers n, m and k ($1 \le n \le 1\,000$, $0 \le m \le 100\,000$, $1 \le k \le n$) — the number of vertices and edges in the graph, and the number of vertices that are homes of the government.

The next line of input will contain k integers $c_1, c_2, ..., c_k$ ($1 \le c_i \le n$). These integers will be pairwise distinct and denote the nodes that are home to the governments in this world.

The following m lines of input will contain two integers u_i and v_i ($1 \le u_i, v_i \le n$). This denotes an undirected edge between nodes u_i and v_i.

It is guaranteed that the graph described by the input is stable.

Output

Output a single integer, the maximum number of edges Hongcow can add to the graph while keeping it stable.

Examples

input

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

output

```
2
```

input

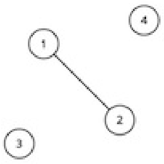
```
3 3 1
2
1 2
1 3
2 3
```

output

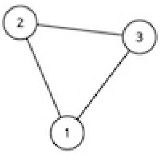
```
0
```

Note

For the first sample test, the graph looks like this:



Vertices 1 and 3 are special. The optimal solution is to connect vertex 4 to vertices 1 and 2. This adds a total of 2 edges. We cannot add any more edges, since vertices 1 and 3 cannot have any path between them.
For the second sample test, the graph looks like this:



We cannot add any more edges to this graph. Note that we are not allowed to add self-loops, and the graph must be simple.

B Police Hypothesis

time limit per test: 8 seconds

memory limit per test: 1024 megabytes

input: standard input

output: standard output

The public transport system of Nlogônia has an express network connecting the main points of interest of the country. There are $N-1$ bullet trains connecting N attractions such that from one of the points of interest you can reach any other using only this network.

As anywhere in the world, it is common that there is graffiti on the train stations. What caught the attention of the police of the country is the fact that in each one of the stations it is possible to find exactly one letter pinned with a specific style. The hypothesis is that criminals may be changing the graffiti as a means of communication and, therefore, it was decided to create a system capable of monitoring the graffiti and its amendments.

Given a pattern P, the description of the connections between stations and the suspicious letters in each of them, your task is to write a program able to deal with the following operations:

- `1` u v: print how many times the pattern P occurs in the path from u to v if we look at the string of characters associated with the consecutive vertices of the path;
- `2` u x: change the suspicious letter at the station u to x.

Input

The first input line contains two integers N and Q ($1 \leq N, Q \leq 10^5$), representing the number of stations and the number of transactions that must be processed. The second line contains the pattern P monitored ($1 \leq |P| \leq 100$). The third line contains a string with N characters representing the letters initially associated with each of the stations. Each of the following $N-1$ lines contains two integers u and v indicating that there is a bullet train between u and v. The following Q lines describe the operations that must be processed as described above.

Output

Your program should print a line for each operation of type 1 containing an integer that represents the number of occurrences of the pattern P on the way.

Examples

input

```
4 4
xtc
xtzy
1 2
2 3
3 4
1 1 3
2 3 c
1 1 3
1 3 1
```

output

```
0
1
0
```

input

```
6 7
lol
dlorlx
1 2
1 3
```

```
3 4
3 5
5 6
1 2 6
2 3 l
2 6 l
2 5 o
1 2 6
2 1 o
1 6 2
```

output

```
0
1
2
```

input

```
5 2
aba
ababa
1 2
2 3
3 4
4 5
1 1 5
1 5 1
```

output

```
2
2
```

C

Conveyor

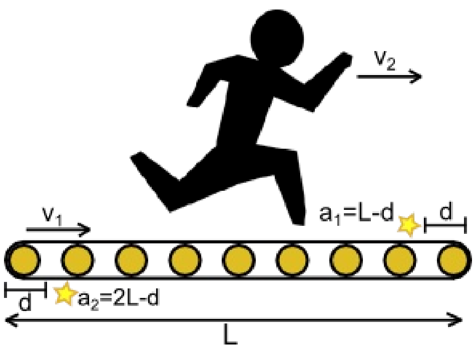
time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Anton came to a chocolate factory. There he found a working conveyor and decided to run on it from the beginning to the end.

The conveyor is a looped belt with a total length of $2l$ meters, of which l meters are located on the surface and are arranged in a straight line. The part of the belt which turns at any moment (the part which emerges from under the floor to the surface and returns from the surface under the floor) is assumed to be negligibly short.

The belt is moving uniformly at speed v_1 meters per second. Anton will be moving on it in the same direction at the constant speed of v_2 meters per second, so his speed relatively to the floor will be $v_1 + v_2$ meters per second. Anton will neither stop nor change the speed or the direction of movement.

Here and there there are chocolates stuck to the belt (n chocolates). They move together with the belt, and do not come off it. Anton is keen on the chocolates, but he is more keen to move forward. So he will pick up all the chocolates he will pass by, but nothing more. If a chocolate is at the beginning of the belt at the moment when Anton starts running, he will take it, and if a chocolate is at the end of the belt at the moment when Anton comes off the belt, he will leave it.



The figure shows an example with two chocolates. One is located in the position $a_1 = l - d$, and is now on the top half of the belt, the second one is in the position $a_2 = 2l - d$, and is now on the bottom half of the belt.

You are given the positions of the chocolates relative to the initial start position of the belt $0 \le a_1 < a_2 < ... < a_n < 2l$. The positions on the belt from 0 to l correspond to the top, and from l to $2l$ — to the the bottom half of the belt (see example). All coordinates are given in meters.

Anton begins to run along the belt at a random moment of time. This means that all possible positions of the belt at the moment he starts running are equiprobable. For each i from 0 to n calculate the probability that Anton will pick up exactly i chocolates.

Input

The first line contains space-separated integers n, l, v_1 and v_2 ($1 \le n \le 10^5$, $1 \le l, v_1, v_2 \le 10^9$) — the number of the chocolates, the length of the conveyor's visible part, the conveyor's speed and Anton's speed.

The second line contains a sequence of space-separated integers $a_1, a_2, ..., a_n$ ($0 \le a_1 < a_2 < ... < a_n < 2l$) — the coordinates of the chocolates.

Output

Print $n + 1$ numbers (one per line): the probabilities that Anton picks up exactly i chocolates, for each i from 0 (the first line) to n (the last line). The answer will be considered correct if each number will have absolute or relative error of at most than 10^{-9}.

Examples

input

```
1 1 1 1
0
```

output

```
0.75000000000000000000
0.25000000000000000000
```

input

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

output

```
0.33333333333333331000
0.66666666666666663000
0.00000000000000000000
```

Note

In the first sample test Anton can pick up a chocolate if by the moment he starts running its coordinate is less than 0.5; but if by the moment the boy starts running the chocolate's coordinate is greater than or equal to 0.5, then Anton won't be able to pick it up. As all positions of the belt are equiprobable, the probability of picking up the chocolate equals $\frac{0.5}{2} = 0.25$, and the probability of not picking it up equals $\frac{1.5}{2} = 0.75$.

D

Bug in Code

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

Recently a serious bug has been found in the FOS code. The head of the F company wants to find the culprit and punish him. For that, he set up an organizational meeting, the issue is: who's bugged the code? Each of the n coders on the meeting said: 'I know for sure that either x or y did it!'

The head of the company decided to choose two suspects and invite them to his office. Naturally, he should consider the coders' opinions. That's why the head wants to make such a choice that at least p of n coders agreed with it. A coder agrees with the choice of two suspects if at least one of the two people that he named at the meeting was chosen as a suspect. In how many ways can the head of F choose two suspects?

Note that even if some coder was chosen as a suspect, he can agree with the head's choice if he named the other chosen coder at the meeting.

Input

The first line contains integers n and p $(3 \le n \le 3 \cdot 10^5; 0 \le p \le n)$ — the number of coders in the F company and the minimum number of agreed people.

Each of the next n lines contains two integers x_i, y_i $(1 \le x_i, y_i \le n)$ — the numbers of coders named by the i-th coder. It is guaranteed that $x_i \ne i,\ y_i \ne i,\ x_i \ne y_i$.

Output

Print a single integer — the number of possible two-suspect sets. Note that the order of the suspects doesn't matter, that is, sets $(1, 2)$ и $(2, 1)$ are considered identical.

Examples

input

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

output

```
6
```

input

```
8 6
5 6
5 7
5 8
6 2
2 1
7 3
1 3
1 4
```

output

```
1
```

E. Two Buttons

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

Vasya has found a strange device. On the front panel of a device there are: a red button, a blue button and a display showing some positive integer. After clicking the red button, device multiplies the displayed number by two. After clicking the blue button, device subtracts one from the number on the display. If at some point the number stops being positive, the device breaks down. The display can show arbitrarily large numbers. Initially, the display shows number n.

Bob wants to get number m on the display. What minimum number of clicks he has to make in order to achieve this result?

Input

The first and the only line of the input contains two distinct integers n and m $(1 \le n, m \le 10^4)$, separated by a space .

Output

Print a single number — the minimum number of times one needs to push the button required to get the number m out of number n.

Examples

input

```
4 6
```

output

```
2
```

input

```
10 1
```

output

```
9
```

Note

In the first example you need to push the blue button once, and then push the red button once.

In the second example, doubling the number is unnecessary, so we need to push the blue button nine times.

F

Looking for Order

time limit per test: 4 seconds

memory limit per test: 512 megabytes

input: standard input

output: standard output

Girl Lena likes it when everything is in order, and looks for order everywhere. Once she was getting ready for the University and noticed that the room was in a mess — all the objects from her handbag were thrown about the room. Of course, she wanted to put them back into her handbag. The problem is that the girl cannot carry more than two objects at a time, and cannot move the handbag. Also, if he has taken an object, she cannot put it anywhere except her handbag — her inherent sense of order does not let her do so.

You are given the coordinates of the handbag and the coordinates of the objects in some Cartesian coordinate system. It is known that the girl covers the distance between any two objects in the time equal to the squared length of the segment between the points of the objects. It is also known that initially the coordinates of the girl and the handbag are the same. You are asked to find such an order of actions, that the girl can put all the objects back into her handbag in a minimum time period.

Input

The first line of the input file contains the handbag's coordinates x_s, y_s. The second line contains number n $(1 \le n \le 24)$ — the amount of objects the girl has. The following n lines contain the objects' coordinates. All the coordinates do not exceed 100 in absolute value. All the given positions are different. All the numbers are integer.

Output

In the first line output the only number — the minimum time the girl needs to put the objects into her handbag.

In the second line output the possible optimum way for Lena. Each object in the input is described by its index number (from 1 to n), the handbag's point is described by number 0. The path should start and end in the handbag's point. If there are several optimal paths, print any of them.

Examples

input

```
0 0
2
1 1
-1 1
```

output

```
8
0 1 2 0
```

input

```
1 1
3
4 3
3 4
0 0
```

output

```
32
0 1 2 0 3 0
```

G. Leaving the bar

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

For a vector $v = (x, y)$, define $|v| = \sqrt{x^2 + y^2}$.

Allen had a bit too much to drink at the bar, which is at the origin. There are n vectors $\vec{v_1}, \vec{v_2}, \cdots, \vec{v_n}$. Allen will make n moves. As Allen's sense of direction is impaired, during the i-th move he will either move in the direction $\vec{v_i}$ or $-\vec{v_i}$. In other words, if his position is currently $p = (x, y)$, he will either move to $p + \vec{v_i}$ or $p - \vec{v_i}$.

Allen doesn't want to wander too far from home (which happens to also be the bar). You need to help him figure out a sequence of moves (a sequence of signs for the vectors) such that his final position p satisfies $|p| \le 1.5 \cdot 10^6$ so that he can stay safe.

Input

The first line contains a single integer n ($1 \le n \le 10^5$) — the number of moves.

Each of the following lines contains two space-separated integers x_i and y_i, meaning that $\vec{v_i} = (x_i, y_i)$. We have that $|v_i| \le 10^6$ for all i.

Output

Output a single line containing n integers $c_1, c_2, \cdots, c_n$, each of which is either 1 or -1. Your solution is correct if the value of $p = \sum_{i=1}^{n} c_i \vec{v_i}$, satisfies $|p| \le 1.5 \cdot 10^6$.

It can be shown that a solution always exists under the given constraints.

Examples

input

```
3
999999 0
0 999999
999999 0
```

output

```
1 1 -1
```

input

```
1
-824590 246031
```

output

```
1
```

input

```
8
-67761 603277
640586 -396671
46147 -122580
569609 -2112
400 914208
131792 309779
-850150 -486293
5272 721899
```

output

```
1 1 1 1 1 1 1 -1
```

H

Sum in the tree

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

Mitya has a rooted tree with n vertices indexed from 1 to n, where the root has index 1. Each vertex v initially had an integer number $a_v \geq 0$ written on it. For every vertex v Mitya has computed s_v: the sum of all values written on the vertices on the path from vertex v to the root, as well as h_v — the depth of vertex v, which denotes the number of vertices on the path from vertex v to the root. Clearly, $s_1 = a_1$ and $h_1 = 1$.

Then Mitya erased all numbers a_v, and by accident he also erased all values s_v for vertices with even depth (vertices with even h_v). Your task is to restore the values a_v for every vertex, or determine that Mitya made a mistake. In case there are multiple ways to restore the values, you're required to find one which minimizes the total sum of values a_v for all vertices in the tree.

Input

The first line contains one integer n — the number of vertices in the tree ($2 \leq n \leq 10^5$). The following line contains integers p_2, p_3, ... p_n, where p_i stands for the parent of vertex with index i in the tree ($1 \leq p_i < i$). The last line contains integer values s_1, s_2, ..., s_n ($-1 \leq s_v \leq 10^9$), where erased values are replaced by -1.

Output

Output one integer — the minimum total sum of all values a_v in the original tree, or -1 if such tree does not exist.

Examples

input

```
5
1 1 1 1
1 -1 -1 -1 -1
```

output

```
1
```

input

```
5
1 2 3 1
1 -1 2 -1 -1
```

output

```
2
```

input

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

output

```
-1
```

I

Multicolored Marbles

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Polycarpus plays with red and blue marbles. He put n marbles from the left to the right in a row. As it turned out, the marbles form a *zebroid*.

A non-empty sequence of red and blue marbles is a *zebroid*, if the colors of the marbles in this sequence alternate. For example, sequences (`red; blue; red`) and (`blue`) are zebroids and sequence (`red; red`) is not a zebroid.

Now Polycarpus wonders, how many ways there are to pick a zebroid **subsequence** from this sequence. Help him solve the problem, find the number of ways modulo 1000000007 (10^9+7).

Input

The first line contains a single integer n $(1 \le n \le 10^6)$ — the number of marbles in Polycarpus's sequence.

Output

Print a single number — the answer to the problem modulo 1000000007 (10^9+7).

Examples

input

```
3
```

output

```
6
```

input

```
4
```

output

```
11
```

Note

Let's consider the first test sample. Let's assume that Polycarpus initially had sequence (`red; blue; red`), so there are six ways to pick a zebroid:

- pick the first marble;
- pick the second marble;
- pick the third marble;
- pick the first and second marbles;
- pick the second and third marbles;
- pick the first, second and third marbles.

It can be proven that if Polycarpus picks (`blue; red; blue`) as the initial sequence, the number of ways won't change.

J

Numbers

time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

Furik loves writing all sorts of problems, especially such that he can't solve himself. You've got one of his problems, the one Furik gave to Rubik. And Rubik asks you to solve it.

There is integer n and array a, consisting of ten integers, indexed by numbers from 0 to 9. Your task is to count the number of positive integers with the following properties:

- the number's length does not exceed n;
- the number doesn't have leading zeroes;
- digit i $(0 \le i \le 9)$ occurs in the number at least $a[i]$ times.

Input

The first line contains integer n $(1 \le n \le 100)$. The next line contains 10 integers $a[0]$, $a[1]$, ..., $a[9]$ $(0 \le a[i] \le 100)$ — elements of array a. The numbers are separated by spaces.

Output

On a single line print the remainder of dividing the answer to the problem by 1000000007 $(10^9 + 7)$.

Examples

input

```
1
0 0 0 0 0 0 0 0 0 1
```

output

```
1
```

input

```
2
1 1 0 0 0 0 0 0 0 0
```

output

```
1
```

input

```
3
1 1 0 0 0 0 0 0 0 0
```

output

```
36
```

Note

In the first sample number 9 meets the requirements.

In the second sample number 10 meets the requirements.

In the third sample numbers **10, 110, 210, 120, 103** meet the requirements. There are other suitable numbers, 36 in total.

K

Neko Performs Cat Furrier Tra

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

Cat Furrier Transform is a popular algorithm among cat programmers to create longcats. As one of the greatest cat programmers ever exist, Neko wants to utilize this algorithm to create the perfect longcat.

Assume that we have a cat with a number x. A perfect longcat is a cat with a number equal $2^m - 1$ for some non-negative integer m. For example, the numbers 0, 1, 3, 7, 15 and so on are suitable for the perfect longcats.

In the Cat Furrier Transform, the following operations can be performed on x:

- (Operation A): you select any non-negative integer n and replace x with $x \oplus (2^n - 1)$, with $\oplus$ being a bitwise XOR operator.
- (Operation B): replace x with $x + 1$.

The first applied operation must be of type A, the second of type B, the third of type A again, and so on. Formally, if we number operations from one in the order they are executed, then odd-numbered operations must be of type A and the even-numbered operations must be of type B.

Neko wants to produce perfect longcats at industrial scale, thus for each cat Neko only wants to perform at most 40 operations. Can you help Neko writing a transformation plan?

Note that it is **not required** to minimize the number of operations. You just need to use no more than 40 operations.

Input

The only line contains a single integer x ($1 \le x \le 10^6$).

Output

The first line should contain a single integer t ($0 \le t \le 40$) — the number of operations to apply.

Then for each odd-numbered operation print the corresponding number n_i in it. That is, print $\lceil \frac{t}{2} \rceil$ integers n_i ($0 \le n_i \le 30$), denoting the replacement x with $x \oplus (2^{n_i} - 1)$ in the corresponding step.

If there are multiple possible answers, you can print any of them. It is possible to show, that there is at least one answer in the constraints of this problem.

Examples

input
```
39
```
output
```
4
5 3
```

input
```
1
```
output
```
0
```

input
7

output
0

Note

In the first test, one of the transforms might be as follows: $39 \to 56 \to 57 \to 62 \to 63$. Or more precisely:

1. Pick $n = 5$. x is transformed into $39 \oplus 31$, or 56.
2. Increase x by 1, changing its value to 57.
3. Pick $n = 3$. x is transformed into $57 \oplus 7$, or 62.
4. Increase x by 1, changing its value to $63 = 2^6 - 1$.

In the second and third test, the number already satisfies the goal requirement.

L

Noldbach problem

time limit per test: 2 seconds

memory limit per test: 64 megabytes

input: standard input

output: standard output

Nick is interested in prime numbers. Once he read about Goldbach problem. It states that every even integer greater than 2 can be expressed as the sum of two primes. That got Nick's attention and he decided to invent a problem of his own and call it Noldbach problem. Since Nick is interested only in prime numbers, Noldbach problem states that at least k prime numbers from 2 to n inclusively can be expressed as the sum of three integer numbers: two neighboring prime numbers and 1. For example, `19 = 7 + 11 + 1`, or `13 = 5 + 7 + 1`.

Two prime numbers are called neighboring if there are no other prime numbers between them.

You are to help Nick, and find out if he is right or wrong.

Input

The first line of the input contains two integers n ($2 \leq n \leq 1000$) and k ($0 \leq k \leq 1000$).

Output

Output `YES` if at least k prime numbers from 2 to n inclusively can be expressed as it was described above. Otherwise output `NO`.

Examples

input

```
27 2
```

output

```
YES
```

input

```
45 7
```

output

```
NO
```

Note

In the first sample the answer is `YES` since at least two numbers can be expressed as it was described (for example, 13 and 19). In the second sample the answer is `NO` since it is impossible to express 7 prime numbers from 2 to 45 in the desired form.