A. Hashing Trees

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

Sasha is taking part in a programming competition. In one of the problems she should check if some rooted trees are isomorphic or not. She has never seen this problem before, but, being an experienced participant, she guessed that she should match trees to some sequences and then compare these sequences instead of trees. Sasha wants to match each tree with a sequence $a_0, a_1, \ldots, a_h$, where h is the height of the tree, and a_i equals to the number of vertices that are at distance of i edges from root.

Unfortunately, this time Sasha's intuition was wrong, and there could be several trees matching the same sequence. To show it, you need to write a program that, given the sequence a_i, builds two non-isomorphic rooted trees that match that sequence, or determines that there is only one such tree.

Two rooted trees are isomorphic, if you can reenumerate the vertices of the first one in such a way, that the index of the root becomes equal the index of the root of the second tree, and these two trees become equal.

The height of a rooted tree is the maximum number of edges on a path from the root to any other vertex.

Input

The first line contains a single integer h $(2 \le h \le 10^5)$ — the height of the tree.

The second line contains $h+1$ integers — the sequence $a_0, a_1, \ldots, a_h$ $(1 \le a_i \le 2 \cdot 10^5)$. The sum of all a_i does not exceed $2 \cdot 10^5$. It is guaranteed that there is at least one tree matching this sequence.

Output

If there is only one tree matching this sequence, print "perfect".

Otherwise print "ambiguous" in the first line. In the second and in the third line print descriptions of two trees in the following format: in one line print $\sum_{i=0}^{h} a_i$ integers, the k-th of them should be the parent of vertex k or be equal to zero, if the k-th vertex is the root.

These treese should be non-isomorphic and should match the given sequence.

Examples

input

```
2
1 1 1
```

output

```
perfect
```

input

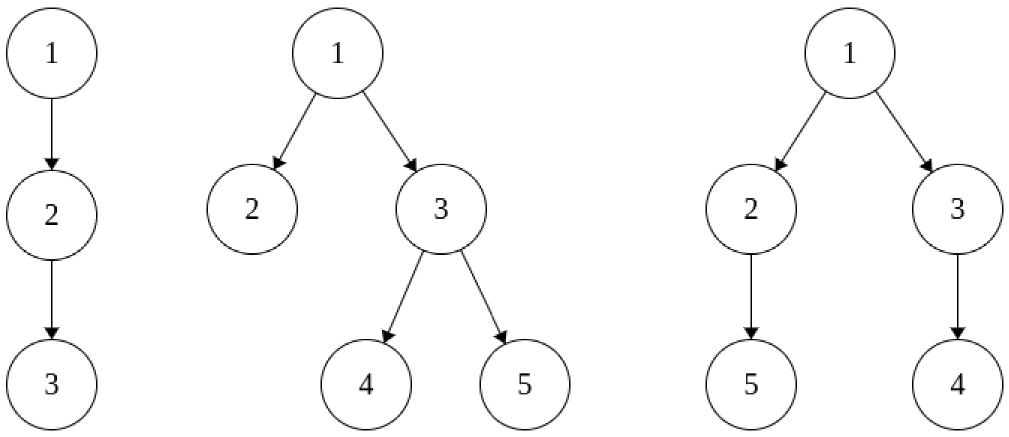
```
2
1 2 2
```

output

```
ambiguous
0 1 1 3 3
0 1 1 3 2
```

Note

The only tree in the first example and the two printed trees from the second example are shown on the picture:


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

B. GCD of Polynomials

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

Suppose you have two polynomials $A(x) = \sum_{k=0}^{n} a_k x^k$ and $B(x) = \sum_{k=0}^{m} b_k x^k$. Then polynomial $A(x)$ can be uniquely represented in the following way:

$$A(x) = B(x) \cdot D(x) + R(x), \deg R(x) < \deg B(x).$$

This can be done using long division. Here, $\deg P(x)$ denotes the degree of polynomial $P(x)$. $R(x)$ is called the remainder of division of polynomial $A(x)$ by polynomial $B(x)$, it is also denoted as $A \bmod B$.

Since there is a way to divide polynomials with remainder, we can define Euclid's algorithm of finding the greatest common divisor of two polynomials. The algorithm takes two polynomials (A, B). If the polynomial $B(x)$ is zero, the result is $A(x)$, otherwise the result is the value the algorithm returns for pair $(B, A \bmod B)$. On each step the degree of the second argument decreases, so the algorithm works in finite number of steps. But how large that number could be? You are to answer this question.

You are given an integer n. You have to build two polynomials with degrees not greater than n, such that their coefficients are integers not exceeding 1 by their absolute value, the leading coefficients (ones with the greatest power of x) are equal to one, and the described Euclid's algorithm performs exactly n steps finding their greatest common divisor. Moreover, the degree of the first polynomial should be greater than the degree of the second. By a step of the algorithm we mean the transition from pair (A, B) to pair $(B, A \bmod B)$.

Input

You are given a single integer n ($1 \le n \le 150$) — the number of steps of the algorithm you need to reach.

Output

Print two polynomials in the following format.

In the first line print a single integer m ($0 \le m \le n$) — the degree of the polynomial.

In the second line print $m + 1$ integers between -1 and 1 — the coefficients of the polynomial, from constant to leading.

The degree of the first polynomial should be greater than the degree of the second polynomial, the leading coefficients should be equal to 1. Euclid's algorithm should perform exactly n steps when called using these polynomials.

If there is no answer for the given n, print `-1`.

If there are multiple answer, print any of them.

Examples

input

```
1
```

output

```
1
0 1
0
1
```

input

```
2
```

output

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

Note

In the second example you can print polynomials $x^2 - 1$ and x. The sequence of transitions is

$$(x^2 - 1, x) \to (x, -1) \to (-1, 0).$$

There are two steps in it.

C. Bipartite Segments

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

You are given an undirected graph with n vertices. There are no edge-simple cycles with the even length in it. In other words, there are no cycles of even length that pass each edge at most once. Let's enumerate vertices from 1 to n.

You have to answer q queries. Each query is described by a segment of vertices $[l; r]$, and you have to count the number of its subsegments $[x; y]$ ($l \le x \le y \le r$), such that if we delete all vertices except the segment of vertices $[x; y]$ (including x and y) and edges between them, the resulting graph is bipartite.

Input

The first line contains two integers n and m ($1 \le n \le 3 \cdot 10^5$, $1 \le m \le 3 \cdot 10^5$) — the number of vertices and the number of edges in the graph.

The next m lines describe edges in the graph. The i-th of these lines contains two integers a_i and b_i ($1 \le a_i, b_i \le n$; $a_i \ne b_i$), denoting an edge between vertices a_i and b_i. It is guaranteed that this graph does not contain edge-simple cycles of even length.

The next line contains a single integer q ($1 \le q \le 3 \cdot 10^5$) — the number of queries.

The next q lines contain queries. The i-th of these lines contains two integers l_i and r_i ($1 \le l_i \le r_i \le n$) — the query parameters.

Output

Print q numbers, each in new line: the i-th of them should be the number of subsegments $[x; y]$ ($l_i \le x \le y \le r_i$), such that the graph that only includes vertices from segment $[x; y]$ and edges between them is bipartite.

Examples

input

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

output

```
5
5
14
```

input

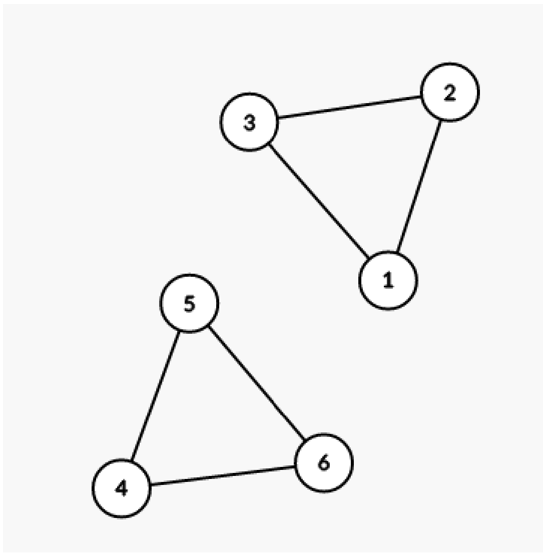
```
8 9
1 2
2 3
3 1
4 5
5 6
6 7
7 8
8 4
7 2
3
1 8
1 4
3 8
```

output

```
27
8
19
```

Note

The first example is shown on the picture below:

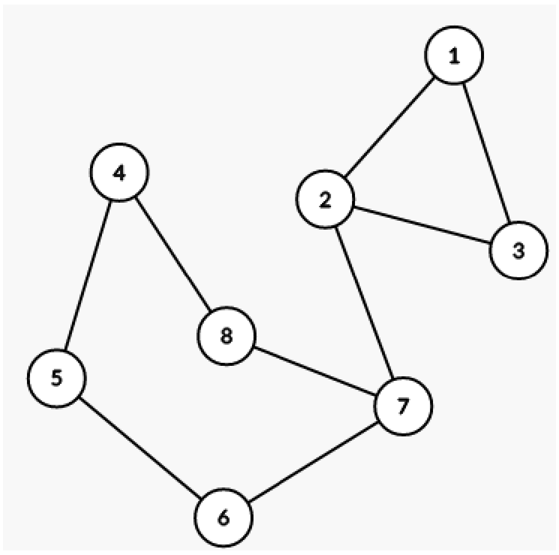


For the first query, all subsegments of $[1; 3]$, except this segment itself, are suitable.

For the first query, all subsegments of $[4; 6]$, except this segment itself, are suitable.

For the third query, all subsegments of $[1; 6]$ are suitable, except $[1; 3]$, $[1; 4]$, $[1; 5]$, $[1; 6]$, $[2; 6]$, $[3; 6]$, $[4; 6]$.

The second example is shown on the picture below:



D. Weighting a Tree

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

You are given a connected undirected graph with n vertices and m edges. The vertices are enumerated from 1 to n.

You are given n integers $c_1, c_2, ..., c_n$, each of them is between $-n$ and n, inclusive. It is also guaranteed that the parity of c_v equals the parity of degree of vertex v. The degree of a vertex is the number of edges connected to it.

You are to write a weight between $-2 \cdot n^2$ and $2 \cdot n^2$ (inclusive) on each edge in such a way, that for each vertex v the sum of weights on edges connected to this vertex is equal to c_v, or determine that this is impossible.

Input

The first line contains two integers n and m ($2 \le n \le 10^5$, $n - 1 \le m \le 10^5$) — the number of vertices and the number of edges.

The next line contains n integers $c_1, c_2, ..., c_n$ ($-n \le c_i \le n$), where c_i is the required sum of weights of edges connected to vertex i. It is guaranteed that the parity of c_i equals the parity of degree of vertex i.

The next m lines describe edges of the graph. The i-th of these lines contains two integers a_i and b_i ($1 \le a_i, b_i \le n$; $a_i \ne b_i$), meaning that the i-th edge connects vertices a_i and b_i.

It is guaranteed that the given graph is connected and does not contain loops and multiple edges.

Output

If there is no solution, print "NO".

Otherwise print "YES" and then m lines, the i-th of them is the weight of the i-th edge w_i ($-2 \cdot n^2 \le w_i \le 2 \cdot n^2$).

Examples

input

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

output

```
YES
1
1
1
```

input

```
4 3
-1 0 2 1
1 2
2 3
3 4
```

output

```
YES
-1
1
1
```

input

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

output

```
YES
3
5
3
-1
-3
5
```

input

```
4 4
```

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

output

```
NO
```

E. Cyclic Cipher

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Senor Vorpal Kickass'o invented an innovative method to encrypt integer sequences of length n. To encrypt a sequence, one has to choose a secret sequence $\{b_i\}_{i=0}^{n-1}$, that acts as a key.

Vorpal is very selective, so the key should be such a sequence b_i, that its cyclic shifts are linearly independent, that is, there is no non-zero set of coefficients $x_0, x_1, \ldots, x_{n-1}$, such that $\sum_{i=0}^{n-1} x_i b_{k-i \bmod n} = 0$ for all k at the same time.

After that for a sequence $\{a_i\}_{i=0}^{n-1}$ you should build the following cipher:

$$c_i = \sum_{k=0}^{n-1} (b_{k-i \bmod n} - a_k)^2$$

In other words, you are to compute the quadratic deviation between each cyclic shift of b_i and the sequence a_i. The resulting sequence is the Kickass's cipher. The cipher is in development right now and Vorpal wants to decipher a sequence after it has been encrypted. You are to solve this problem for him. You are given sequences c_i and b_i. You are to find all suitable sequences a_i.

Input

The first line contains a single integer n ($1 \le n \le 10^5$).

The second line contains n integers $b_0, b_1, \ldots, b_{n-1}$ ($0 \le b_i \le 10^3$).

The third line contains n integers $c_0, c_1, \ldots, c_{n-1}$ ($0 \le c_i \le 5 \cdot 10^6$).

It is guaranteed that all cyclic shifts of sequence b_i are linearly independent.

Output

In the first line print a single integer k — the number of sequences a_i, such that after encrypting them with key b_i you get the sequence c_i.

After that in each of k next lines print n integers $a_0, a_1, \ldots, a_{n-1}$. Print the sequences in lexicographical order.

Note that k could be equal to 0.

Examples

input

```
1
1
0
```

output

```
1
1
```

input

```
1
100
81
```

output

```
2
91
109
```

input

```
3
1 1 3
165 185 197
```

output

```
2
-6 -9 -1
8 5 13
```