Codeforces Global Round 16

A. Median Maximization

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

You are given two positive integers n and s. Find the maximum possible median of an array of n **non-negative** integers (not necessarily distinct), such that the sum of its elements is equal to s.

A **median** of an array of integers of length m is the number standing on the $\lceil \frac{m}{2} \rceil$-th (rounding up) position in the non-decreasing ordering of its elements. Positions are numbered starting from 1. For example, a median of the array $[20, 40, 20, 50, 50, 30]$ is the $\lceil \frac{m}{2} \rceil$-th element of $[20, 20, 30, 40, 50, 50]$, so it is 30. There exist other definitions of the median, but in this problem we use the described definition.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. Description of the test cases follows.

Each test case contains a single line with two integers n and s ($1 \le n, s \le 10^9$) — the length of the array and the required sum of the elements.

Output

For each test case print a single integer — the maximum possible median.

Example

input
8 1 5 2 5 3 5 2 1 7 17 4 14 1 1000000000 1000000000 1
output
5 2 2 0 4 4 1000000000 0

Note

Possible arrays for the first three test cases (in each array the median is underlined):

- In the first test case $[\underline{5}]$
- In the second test case $[\underline{2}, 3]$
- In the third test case $[1, \underline{2}, 2]$

B. MIN-MEX Cut

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

A binary string is a string that consists of characters 0 and 1.

Let MEX of a binary string be the smallest digit among 0, 1, or 2 that does not occur in the string. For example, MEX of 001011 is 2, because 0 and 1 occur in the string at least once, MEX of 1111 is 0, because 0 and 2 do not occur in the string and $0 < 2$.

A binary string s is given. You should cut it into any number of substrings such that each character is in exactly one substring. It is possible to cut the string into a single substring — the whole string.

A string a is a substring of a string b if a can be obtained from b by deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

What is the **minimal** sum of MEX of all substrings pieces can be?

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. Description of the test cases follows.

Each test case contains a single binary string s ($1 \le |s| \le 10^5$).

It's guaranteed that the sum of lengths of s over all test cases does not exceed 10^5.

Output

For each test case print a single integer — the minimal sum of MEX of all substrings that it is possible to get by cutting s optimally.

Example

input
6 01

```
1111
01100
101
0000
01010
```

output
1 0 2 1 1 2

Note

In the first test case the minimal sum is $\text{MEX}(0) + \text{MEX}(1) = 1 + 0 = 1$.

In the second test case the minimal sum is $\text{MEX}(1111) = 0$.

In the third test case the minimal sum is $\text{MEX}(01100) = 2$.

C. MAX-MEX Cut

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

A binary string is a string that consists of characters 0 and 1. A bi-table is a table that has exactly two rows of equal length, each being a binary string.

Let MEX of a bi-table be the smallest digit among 0, 1, or 2 that does not occur in the bi-table. For example, MEX for $\begin{bmatrix} 0011 \\ 1010 \end{bmatrix}$ is 2, because 0 and 1 occur in the bi-table at least once. MEX for $\begin{bmatrix} 111 \\ 111 \end{bmatrix}$ is 0, because 0 and 2 do not occur in the bi-table, and $0 < 2$.

You are given a bi-table with n columns. You should cut it into any number of bi-tables (each consisting of consecutive columns) so that each column is in exactly one bi-table. It is possible to cut the bi-table into a single bi-table — the whole bi-table.

What is the **maximal** sum of MEX of all resulting bi-tables can be?

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. Description of the test cases follows.

The first line of the description of each test case contains a single integer n ($1 \le n \le 10^5$) — the number of columns in the bi-table.

Each of the next two lines contains a binary string of length n — the rows of the bi-table.

It's guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case print a single integer — the maximal sum of MEX of all bi-tables that it is possible to get by cutting the given bi-table optimally.

Example

```
4
7
0101000
1101100
5
01100
10101
2
01
01
6
000000
111111
```

output
8 8 2 12

Note

In the first test case you can cut the bi-table as follows:

- $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$, its MEX is 2.
- $\begin{bmatrix} 10 \\ 10 \end{bmatrix}$, its MEX is 2.
- $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$, its MEX is 0.
- $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$, its MEX is 2.
- $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$, its MEX is 1.
- $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$, its MEX is 1.

The sum of MEX is 8.

D1. Seating Arrangements (easy version)

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

It is the easy version of the problem. The only difference is that in this version $n = 1$.

In the cinema seats can be represented as the table with n rows and m columns. The rows are numbered with integers from 1 to n. The seats in each row are numbered with consecutive integers from left to right: in the k-th row from $m(k-1)+1$ to mk for all rows $1 \le k \le n$.

1	2	$\cdots$	$m-1$	m
$m+1$	$m+2$	$\cdots$	$2m-1$	$2m$
$2m+1$	$2m+2$	$\cdots$	$3m-1$	$3m$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$m(n-1)+1$	$m(n-1)+2$	$\cdots$	$nm-1$	nm

The table with seats indices

There are nm people who want to go to the cinema to watch a new film. They are numbered with integers from 1 to nm. You should give exactly one seat to each person.

It is known, that in this cinema as lower seat index you have as better you can see everything happening on the screen. i-th person has the level of sight a_i. Let's define s_i as the seat index, that will be given to i-th person. You want to give better places for people with lower sight levels, so for any two people i, j such that $a_i < a_j$ it should be satisfied that $s_i < s_j$.

After you will give seats to all people they will start coming to their seats. In the order from 1 to nm, each person will enter the hall and sit in their seat. To get to their place, the person will go to their seat's row and start moving from the first seat in this row to theirs from left to right. While moving some places will be free, some will be occupied with people already seated. The **inconvenience** of the person is equal to the number of occupied seats he or she will go through.

Let's consider an example: $m = 5$, the person has the seat 4 in the first row, the seats $1, 3, 5$ in the first row are already occupied, the seats 2 and 4 are free. The inconvenience of this person will be 2, because he will go through occupied seats 1 and 3.

Find the minimal total inconvenience (the sum of inconveniences of all people), that is possible to have by giving places for all people (all conditions should be satisfied).

Input
The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases. Description of the test cases follows.

The first line of each test case contains two integers n and m ($n = 1$, $1 \le m \le 300$) — the number of rows and places in each row respectively.

The second line of each test case contains $n \cdot m$ integers $a_1, a_2, \ldots, a_{n \cdot m}$ ($1 \le a_i \le 10^9$), where a_i is the sight level of i-th person.

It's guaranteed that the sum of $n \cdot m$ over all test cases does not exceed 10^5.

Output
For each test case print a single integer — the minimal total inconvenience that can be achieved.

Example

input
4 1 3 1 2 3 1 5 2 1 5 3 3 1 2 2 1 1 6 2 3 2 1 1 1
output
3 6 0 1

Note
In the first test case, there is a single way to arrange people, because all sight levels are distinct. The first person will sit on the first seat, the second person will sit on the second place, the third person will sit on the third place. So inconvenience of the first person will be 0, inconvenience of the second person will be 1 and inconvenience of the third person will be 2. The total inconvenience is $0 + 1 + 2 = 3$.

In the second test case, people should sit as follows: $s_1 = 2$, $s_2 = 1$, $s_3 = 5$, $s_4 = 4$, $s_5 = 3$. The total inconvenience will be 6.

D2. Seating Arrangements (hard version)

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

It is the hard version of the problem. The only difference is that in this version $1 \le n \le 300$.

In the cinema seats can be represented as the table with n rows and m columns. The rows are numbered with integers from 1 to n. The seats in each row are numbered with consecutive integers from left to right: in the k-th row from $m(k-1)+1$ to mk for all rows $1 \le k \le n$.

1	2	$\cdots$	$m-1$	m

$m+1$	$m+2$	$\cdots$	$2m-1$	$2m$
$2m+1$	$2m+2$	$\cdots$	$3m-1$	$3m$
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
$m(n-1)+1$	$m(n-1)+2$	$\cdots$	$nm-1$	nm

The table with seats indices

There are nm people who want to go to the cinema to watch a new film. They are numbered with integers from 1 to nm. You should give exactly one seat to each person.

It is known, that in this cinema as lower seat index you have as better you can see everything happening on the screen. i-th person has the level of sight a_i. Let's define s_i as the seat index, that will be given to i-th person. You want to give better places for people with lower sight levels, so for any two people i, j such that $a_i < a_j$ it should be satisfied that $s_i < s_j$.

After you will give seats to all people they will start coming to their seats. In the order from 1 to nm, each person will enter the hall and sit in their seat. To get to their place, the person will go to their seat's row and start moving from the first seat in this row to theirs from left to right. While moving some places will be free, some will be occupied with people already seated. The **inconvenience** of the person is equal to the number of occupied seats he or she will go through.

Let's consider an example: $m = 5$, the person has the seat 4 in the first row, the seats 1, 3, 5 in the first row are already occupied, the seats 2 and 4 are free. The inconvenience of this person will be 2, because he will go through occupied seats 1 and 3.

Find the minimal total inconvenience (the sum of inconveniences of all people), that is possible to have by giving places for all people (all conditions should be satisfied).

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases. Description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n, m \le 300$) — the number of rows and places in each row respectively.

The second line of each test case contains $n \cdot m$ integers $a_1, a_2, \ldots, a_{n \cdot m}$ ($1 \le a_i \le 10^9$), where a_i is the sight level of i-th person.

It's guaranteed that the sum of $n \cdot m$ over all test cases does not exceed 10^5.

Output

For each test case print a single integer — the minimal total inconvenience that can be achieved.

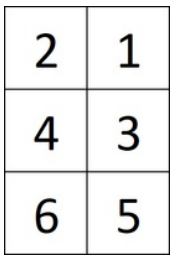
Example

input
7 1 2 1 2 3 2 1 1 2 2 3 3 3 3 3 4 4 1 1 1 1 1 2 2 2 1 1 2 1 4 2 50 50 50 50 3 50 50 50 4 2 6 6 6 6 2 2 9 6 2 9 1 3 3 3 3 3 1 1 3 1 3 1 1 3 3 1 1 3
output
1 0 4 0 0 0 1

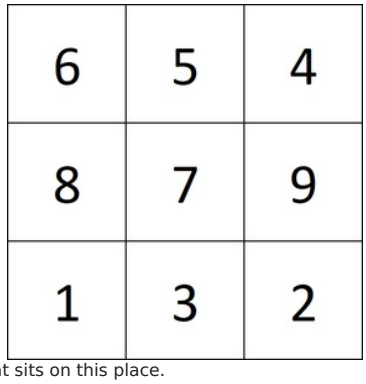
Note

In the first test case, there is a single way to give seats: the first person sits in the first place and the second person — in the second. The total inconvenience is 1.

In the second test case the optimal seating looks like this:



In the third test case the optimal seating looks like this:



The number in a cell is the person's index that sits on this place.

E. Buds Re-hanging

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

A tree is a connected graph without cycles. A rooted tree has a special vertex called the root. The parent of a vertex v (different from root) is the previous to v vertex on the shortest path from the root to the vertex v. Children of the vertex v are all vertices for which v is the parent.

A vertex is a leaf if it has no children. We call a vertex a **bud**, if the following three conditions are satisfied:

- it is not a root,
- it has at least one child, and
- all its children are leaves.

You are given a rooted tree with n vertices. The vertex 1 is the root. In one operation you can choose any bud with all its children (they are leaves) and re-hang them to any other vertex of the tree. By doing that you delete the edge connecting the bud and its parent and add an edge between the bud and the chosen vertex of the tree. The chosen vertex cannot be the bud itself or any of its children. All children of the bud stay connected to the bud.

What is the minimum number of leaves it is possible to get if you can make any number of the above-mentioned operations (possibly zero)?

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. Description of the test cases follows.

The first line of each test case contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the number of the vertices in the given tree.

Each of the next $n - 1$ lines contains two integers u and v ($1 \le u, v \le n$, $u \ne v$) meaning that there is an edge between vertices u and v in the tree.

It is guaranteed that the given graph is a tree.

It is guaranteed that the sum of n over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case print a single integer — the minimal number of leaves that is possible to get after some operations.

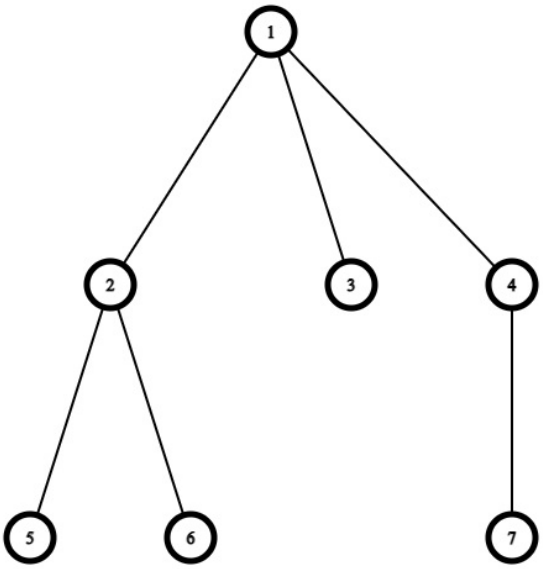
Example

input
5 7 1 2 1 3 1 4 2 5 2 6 4 7 6 1 2 1 3 2 4 2 5 3 6 2 1 2 7 7 3 1 5 1 3 4 6 4 7 2 1 6 2 1 2 3 4 5 3 4 3 6

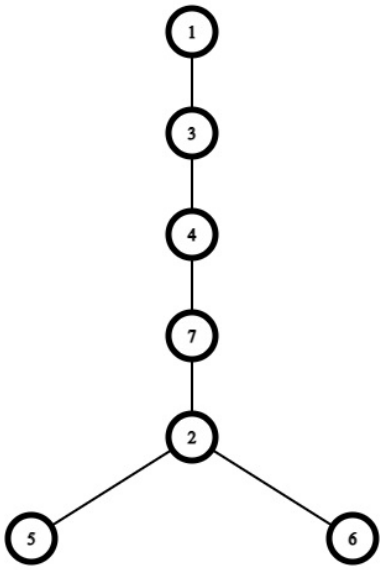
output
2 2 1 2 1

Note

In the first test case the tree looks as follows:

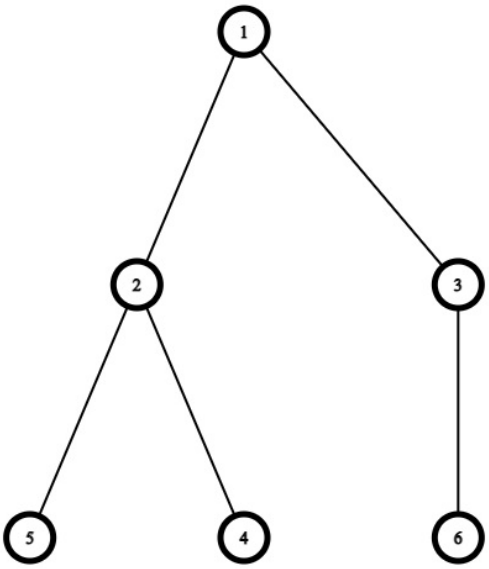


Firstly you can choose a bud vertex 4 and re-hang it to vertex 3. After that you can choose a bud vertex 2 and re-hang it to vertex 7. As a result, you will have the following tree with 2 leaves:

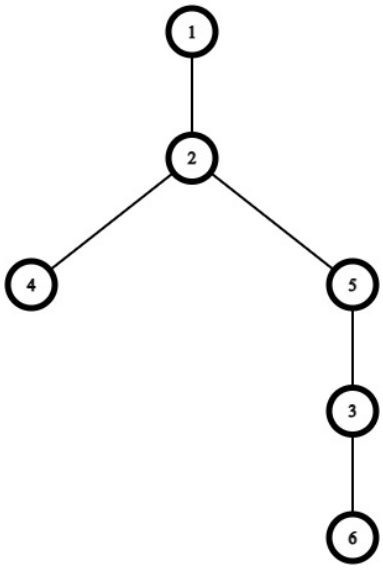


It can be proved that it is the minimal number of leaves possible to get.

In the second test case the tree looks as follows:



You can choose a bud vertex 3 and re-hang it to vertex 5. As a result, you will have the following tree with 2 leaves:



It can be proved that it is the minimal number of leaves possible to get.

F. Points Movement

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

There are n points and m segments on the coordinate line. The initial coordinate of the i-th point is a_i. The endpoints of the j-th segment are l_j and r_j — left and right endpoints, respectively.

You can move the points. In one move you can move any point from its current coordinate x to the coordinate $x - 1$ or the coordinate $x + 1$. The cost of this move is 1.

You should move the points in such a way that each segment is visited by at least one point. A point visits the segment $[l, r]$ if there is a moment when its coordinate was on the segment $[l, r]$ (including endpoints).

You should find the minimal possible total cost of all moves such that all segments are visited.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. Description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the number of points and segments respectively.

The next line contains n **distinct** integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$) — the initial coordinates of the points.

Each of the next m lines contains two integers l_j, r_j ($-10^9 \le l_j \le r_j \le 10^9$) — the left and the right endpoints of the j-th segment.

It's guaranteed that the sum of n and the sum of m over all test cases does not exceed $2 \cdot 10^5$.

Output
For each test case print a single integer — the minimal total cost of all moves such that all segments are visited.

Example

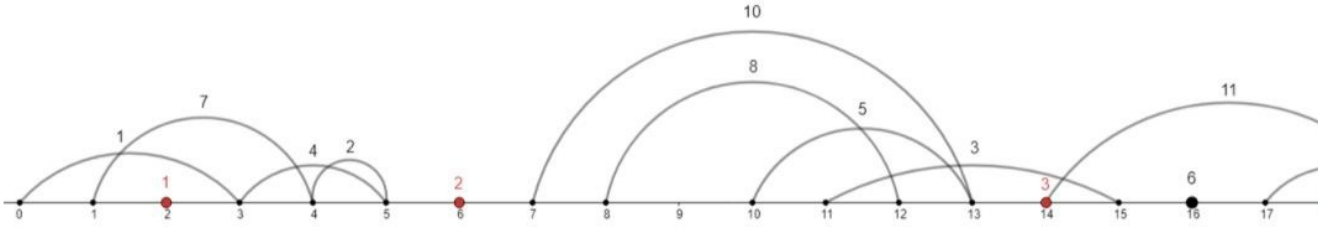
input
2 4 11 2 6 14 18 0 3 4 5 11 15 3 5 10 13 16 16 1 4 8 12 17 19 7 13 14 19 4 12 -9 -16 12 3 -20 -18 -14 -13 -10 -7 -3 -1 0 4 6 11 7 9 8 10 13 15 14 18 16 17 18 19
output
5 22

Note
In the first test case the points can be moved as follows:

- Move the second point from the coordinate 6 to the coordinate 5.
- Move the third point from the coordinate 14 to the coordinate 13.
- Move the fourth point from the coordinate 18 to the coordinate 17.
- Move the third point from the coordinate 13 to the coordinate 12.
- Move the fourth point from the coordinate 17 to the coordinate 16.

The total cost of moves is 5. It is easy to see, that all segments are visited by these movements. For example, the tenth segment ($[7, 13]$) is visited after the second move by the third point.

Here is the image that describes the first test case:



G. Four Vertices

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

You are given an undirected weighted graph, consisting of n vertices and m edges.

Some queries happen with this graph:

- Delete an existing edge from the graph.
- Add a non-existing edge to the graph.

At the beginning and after each query, you should find four **different** vertices a, b, c, d such that there exists a path between a and b, there exists a path between c and d, and the sum of lengths of two shortest paths from a to b and from c to d is minimal. The answer to the query is the sum of the lengths of these two shortest paths. The length of the path is equal to the sum of weights of edges in this path.

Input
The first line contains two integers n and m $(4 \leq n, m \leq 10^5)$ — the number of vertices and edges in the graph respectively.

Each of the next m lines contain three integers v, u, w $(1 \leq v, u \leq n, v \neq u, 1 \leq w \leq 10^9)$ — this triple means that there is an edge between vertices v and u with weight w.

The next line contains a single integer q $(0 \leq q \leq 10^5)$ — the number of queries.

The next q lines contain the queries of two types:

- 0 v u — this query means deleting an edge between v and u $(1 \leq v, u \leq n, v \neq u)$. It is guaranteed that such edge exists in the graph.
- 1 v u w — this query means adding an edge between vertices v and u with weight w $(1 \leq v, u \leq n, v \neq u, 1 \leq w \leq 10^9)$. It is guaranteed that there was no such edge in the graph.

It is guaranteed that the initial graph does not contain multiple edges.

At the beginning and after each query, the graph **doesn't need** to be connected.

It is guaranteed that at each moment the number of edges **will be at least** 4. It can be proven, that at each moment there exist some four vertices a, b, c, d such that there exists a path between vertices a and b, and there exists a path between vertices c and d.

Output
Print $q + 1$ integers — the minimal sum of lengths of shortest paths between chosen pairs of vertices before the queries and after each of them.

Example

input
6 6 1 3 6 4 3 1 1 4 1 2 6 4 2 4 2 5 4 3 4 1 2 5 2 0 1 4 0 3 4 1 6 1 3
output
4 3 3 7 5

Note
Before the queries you can choose vertices $(a, b) = (3, 2)$ and $(c, d) = (1, 4)$. The sum of lengths of two shortest paths is $3 + 1 = 4$.

After the first query you can choose vertices $(a, b) = (2, 5)$ and $(c, d) = (1, 4)$. The sum of lengths of two shortest paths is $2 + 1 = 3$.

After the second query you can choose vertices $(a, b) = (3, 4)$ and $(c, d) = (2, 5)$. The sum of lengths of two shortest paths is $1 + 2 = 3$.

After the third query, you can choose vertices $(a, b) = (2, 6)$ and $(c, d) = (4, 5)$. The sum of lengths of two shortest paths is $4 + 3 = 7$.

After the last query you can choose vertices $(a, b) = (1, 6)$ and $(c, d) = (2, 5)$. The sum of lengths of two shortest paths is $3 + 2 = 5$.

H. Xor-quiz

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

This is an interactive problem.

You are given two integers c and n. The jury has a **randomly generated** set A of distinct positive integers not greater than c (it is generated from all such possible sets with equal probability). The size of A is equal to n.

Your task is to guess the set A. In order to guess it, you can ask at most $\lceil 0.65 \cdot c \rceil$ queries.

In each query, you choose a single integer $1 \le x \le c$. As the answer to this query you will be given the bitwise xor sum of all y, such that $y \in A$ and $gcd(x, y) = 1$ (i.e. x and y are coprime). If there is no such y this xor sum is equal to 0.

You can ask all queries at the beginning and you will receive the answers to all your queries. After that, you won't have the possibility to ask queries.

You should find any set A', such that $|A'| = n$ and A' and A have the same answers for all c possible queries.

Input
Firstly you are given two integers c and n ($100 \le c \le 10^6$, $0 \le n \le c$).

Interaction
In the first line you should print an integer q ($0 \le q \le \lceil 0.65 \cdot c \rceil$) — the number of queries you want to ask. After that in the same line print q integers $x_1, x_2, \ldots, x_q$ ($1 \le x_i \le c$) — the queries.

For these queries you should read q integers, i-th of them is the answer to the described query for $x = x_i$.

After that you should print n distinct integers $A'_1, A'_2, \ldots, A'_n$ — the set A' you found.

If there are different sets A' that have the same answers for all possible queries, print any of them.

If you will ask more than $\lceil 0.65 \cdot c \rceil$ queries or if the queries will be invalid, the interactor will terminate immediately and your program will receive verdict `Wrong Answer`.

After printing the queries and answers do not forget to output end of line and flush the output buffer. Otherwise, you will get the `Idleness limit exceeded` verdict. To do flush use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- Read documentation for other languages.

Hacks

You cannot make hacks in this problem.

Example

input
10 6 1 4 2 11 4 4 4

output
7 10 2 3 5 7 1 6 1 4 5 6 8 10

Note

The sample is made only for you to understand the interaction protocol. **Your solution will not be tested on the sample.**

In the sample $A = \{1, 4, 5, 6, 8, 10\}$. 7 queries are made, $7 \leq \lceil 0.65 \cdot 10 \rceil = 7$, so the query limit is not exceeded.

Answers for the queries:

- For 10: 1 is the only number in the set A coprime with 10, so the answer is 1
- For 2: $1_{10} \oplus 5_{10} = 001_2 \oplus 101_2 = 4_{10}$, where $\oplus$ is the bitwise xor
- For 3: $1_{10} \oplus 4_{10} \oplus 5_{10} \oplus 8_{10} \oplus 10_{10} = 0001_2 \oplus 0100_2 \oplus 0101_2 \oplus 1000_2 \oplus 1010_2 = 2_{10}$
- For 5: $1_{10} \oplus 4_{10} \oplus 6_{10} \oplus 8_{10} = 0001_2 \oplus 0100_2 \oplus 0110_2 \oplus 1000_2 = 11_{10}$
- For 7: $1_{10} \oplus 4_{10} \oplus 5_{10} \oplus 6_{10} \oplus 8_{10} \oplus 10_{10} = 0001_2 \oplus 0100_2 \oplus 0101_2 \oplus 0110_2 \oplus 1000_2 \oplus 1010_2 = 4_{10}$
- For 1: $1_{10} \oplus 4_{10} \oplus 5_{10} \oplus 6_{10} \oplus 8_{10} \oplus 10_{10} = 0001_2 \oplus 0100_2 \oplus 0101_2 \oplus 0110_2 \oplus 1000_2 \oplus 1010_2 = 4_{10}$
- For 6: $1_{10} \oplus 5_{10} = 0001_2 \oplus 0101_2 = 4_{10}$