A. Two 0-1 Sequences

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

AquaMoon has two binary sequences a and b, which contain only 0 and 1. AquaMoon can perform the following two operations any number of times (a_1 is the first element of a, a_2 is the second element of a, and so on):

- *Operation 1*: if a contains at least two elements, change a_2 to $\min(a_1, a_2)$, and remove the first element of a.
- *Operation 2*: if a contains at least two elements, change a_2 to $\max(a_1, a_2)$, and remove the first element of a.

Note that after a removal of the first element of a, the former a_2 becomes the first element of a, the former a_3 becomes the second element of a and so on, and the length of a reduces by one.

Determine if AquaMoon can make a equal to b by using these operations.

Input

The first line contains a single integer t ($1 \leq t \leq 2\,000$) — the number of test cases. Description of test cases follows.

The first line of each test case contains two integers n, m ($1 \leq n, m \leq 50$, $m \leq n$) — the lengths of a and b respectively.

The second line of each test case contains a string a of length n, consisting only 0 and 1.

The third line of each test case contains a string b of length m, consisting only 0 and 1.

Output

For each test case, output "YES" if AquaMoon can change a to b by using these options; otherwise, output "NO".

You may print each letter in any case (for example, "YES", "Yes", "yes", "yEs" will all be recognized as a positive answer).

Example

input
10 6 2 001001 11 6 2 110111 01 6 2 000001 11 6 2 111111 01 8 5 10000101 11010 7 4 1010001 1001 8 6 01010010 010010 8 4 01010101 1001 8 4 10101010 0110 7 5 1011100 11100
output
YES YES NO NO NO YES YES NO NO YES

Note
In the first test case, you can use *Operation 2* four times to make a equals to b.

In the second test case, you can use *Operation 1* four times to make a equals to b.

In the third test case, it can be proved that no matter how we use the operations, it is impossible to make a equal to b.

In the fourth test case, it can be proved that no matter how we use the operations, it is impossible to make a equal to b.

In the fifth test case, you can use *Operation 2* three times to make a become 10101, so the first element of a equals to the first element of b, but it can be proved that no matter how to operate, the second to the fifth elements of a can't be the same as b.

B. Luke is a Foodie

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

Luke likes to eat. There are n piles of food aligned in a straight line in front of him. The i-th pile contains a_i units of food.

Luke will walk from the 1-st pile towards the n-th pile, and he wants to eat every pile of food without walking back. When Luke reaches the i-th pile, he can eat that pile if and only if $|v - a_i| \leq x$, where x is a fixed integer, and v is Luke's food affinity.

Before Luke starts to walk, he can set v to any integer. Also, for each i ($1 \leq i \leq n$), Luke can *change* his food affinity to any integer **before** he eats the i-th pile.

Find the minimum number of *changes* needed to eat every pile of food.

Note that the initial choice for v is **not** considered as a change.

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

For each test case, the first line contains two integers, n, x ($1 \leq n \leq 2 \cdot 10^5$, $1 \leq x \leq 10^9$) — the number of piles, and the maximum difference between the size of a pile and Luke's food affinity, such that Luke can eat the pile.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$).

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

Output
For each test case, output an integer on a separate line, which is the minimum number of changes needed.

Example

input
7 5 3 3 8 5 6 7 5 3 3 10 9 8 7 12 8 25 3 3 17 8 6 1 16 15 25 17 23 10 2 1 2 3 4 5 6 7 8 9 10 8 2 2 4 6 8 6 4 12 14 8 2 2 7 8 9 6 13 21 28 15 5 11 4 13 23 7 10 5 21 20 11 17 5 29 16 11
output
0 1 2 1 2 4 6

Note
In the first test case, Luke can set v to 5 before he starts to walk. And he can walk straight to eat every piles of food without changing v.

In the second test case, Luke can set v to 3 before he starts to walk. And he could change v to 10 before he eats the second pile. After that, he can walk straight to eat remaining food without changing v.

In the fourth test case, Luke can set v to 3 before he starts to walk. And he could change v to 8 before he eats the sixth pile. After that, he can walk straight to eat remaining food without changing v.

In the fifth test case, Luke can set v to 4 before he starts to walk. And he could change v to 6 before he eats the fourth pile. Then he could change v to 12 before he eats the seventh pile. After that, he can walk straight to eat remaining food without changing v.

C. Virus

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

There are n houses numbered from 1 to n on a circle. For each $1 \leq i \leq n-1$, house i and house $i+1$ are neighbours; additionally, house n and house 1 are also neighbours.

Initially, m of these n houses are infected by a deadly virus. Each **morning**, Cirno can choose a house which is uninfected and protect the house from being infected permanently.

Every day, the following things happen in order:

- Cirno chooses an uninfected house, and protect it permanently.
- All uninfected, unprotected houses which have at least one **infected** neighbor become infected.

Cirno wants to stop the virus from spreading. Find the minimum number of houses that will be infected in the end, if she optimally choose the houses to protect.

Note that every day Cirno always chooses a house to protect **before** the virus spreads. Also, a protected house will not be infected forever.

Input

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

The first line of each test case consists of two positive integers n, m ($5 \leq n \leq 10^9$, $1 \leq m \leq \min(n, 10^5)$) — the number of houses on the circle, and the number of houses that are initially infected.

The second line of each test case consists of m distinct positive integers $a_1, a_2, \cdots, a_m$ ($1 \leq a_i \leq n$) — the indices of the houses infected initially.

It is guaranteed that the sum of m over all test cases does not exceed 10^5.

Output

For each test case, output an integer on a separate line, which is the minimum number of infected houses in the end.

Example

input
8 10 3 3 6 8 6 2 2 5 20 3 3 7 12 41 5 1 11 21 31 41 10 5 2 4 6 8 10 5 5 3 2 5 4 1 1000000000 1 1 1000000000 4 1 1000000000 10 16
output
7 5 11 28 9 5 2 15

Note

In the first test case:

At the start of the first day, house 3, 6, 8 are infected. Choose house 2 to protect.

At the start of the second day, house 3, 4, 5, 6, 7, 8, 9 are infected. Choose house 10 to protect.

At the start of the third day, no more houses are infected.

In the second test case:

At the start of the first day, house 2, 5 are infected. Choose house 1 to protect.

At the start of the second day, house 2, 3, 4, 5, 6 are infected. No more available houses can be protected.

D. Magical Array

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

Eric has an array b of length m, then he generates n additional arrays $c_1, c_2, \ldots, c_n$, each of length m, from the array b, by the following way:

Initially, $c_i = b$ for every $1 \leq i \leq n$. Eric secretly chooses an integer k $(1 \leq k \leq n)$ and chooses c_k to be the special array.

There are two operations that Eric can perform on an array c_t:

- *Operation 1*: Choose two integers i and j $(2 \leq i < j \leq m-1)$, subtract 1 from both $c_t[i]$ and $c_t[j]$, and add 1 to both $c_t[i-1]$ and $c_t[j+1]$. **That operation can only be used on a non-special array, that is when $t \neq k$.**;
- *Operation 2*: Choose two integers i and j $(2 \leq i < j \leq m-2)$, subtract 1 from both $c_t[i]$ and $c_t[j]$, and add 1 to both $c_t[i-1]$ and $c_t[j+2]$. **That operation can only be used on a special array, that is when $t = k$.**
  Note that Eric can't perform an operation if any element of the array will become less than 0 after that operation.

Now, Eric does the following:

- For every **non-special** array c_i $(i \neq k)$, Eric uses **only operation 1** on it **at least once**.
- For the **special** array c_k, Eric uses **only operation 2** on it **at least once**.

Lastly, Eric discards the array b.

For given arrays $c_1, c_2, \ldots, c_n$, your task is to find out the special array, i.e. the value k. Also, you need to find the number of times of operation 2 was used on it.

Input

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

The first line of each test case contains two integers n and m $(3 \leq n \leq 10^5, 7 \leq m \leq 3 \cdot 10^5)$ — the number of arrays given to you, and the length of each array.

The next n lines contains m integers each, $c_{i,1}, c_{i,2}, \ldots, c_{i,m}$.

It is guaranteed that each element of the discarded array b is in the range $[0, 10^6]$, and therefore $0 \leq c_{i,j} \leq 3 \cdot 10^{11}$ for all possible pairs of (i, j).

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed 10^6.

It is guaranteed that the input is generated according to the procedure above.

Output

For each test case, output one line containing two integers — the index of the special array, and the number of times that *Operation 2* was performed on it. It can be shown that under the constraints given in the problem, this value is unique and won't exceed 10^{18}, so you can represent it as a 64-bit integer. It can also be shown that the index of the special array is uniquely determined.

In this problem, **hacks are disabled**.

Example

input
7 3 9 0 1 2 0 0 2 1 1 0 0 1 1 1 2 0 0 2 0 0 1 2 0 0 1 2 1 0 3 7 25 15 20 15 25 20 20 26 14 20 14 26 20 20 25 15 20 15 20 20 25 3 9 25 15 20 15 25 20 20 20 20 26 14 20 14 26 20 20 20 20 25 15 20 15 25 15 20 20 25 3 11 25 15 20 15 25 20 20 20 20 20 20 26 14 20 14 26 20 20 20 20 20 20 25 15 20 15 25 20 15 20 20 20 25 3 13 25 15 20 15 25 20 20 20 20 20 20 20 20 26 14 20 14 26 20 20 20 20 20 20 20 20 25 15 20 15 25 20 20 15 20 20 20 20 25 3 15 25 15 20 15 25 20 20 20 20 20 20 20 20 20 20

```
26 14 20 14 26 20 20 20 20 20 20 20 20 20 20
25 15 20 15 25 20 20 20 15 20 20 20 20 20 25
3 9
909459 479492 676924 224197 162866 164495 193268 742456 728277
948845 455424 731850 327890 304150 237351 251763 225845 798316
975446 401170 792914 272263 300770 242037 236619 334316 725899
```

output

```
3 1
3 10
3 15
3 20
3 25
3 30
1 1378716
```

Note

In the first test case, the secret array b is $[0, 1, 1, 1, 1, 1, 1, 0]$. Array c_1 and array c_2 are generated by using operation 1. Array c_3 is generated by using operation 2.

For Array c_1,you can choose $i = 4$ and $j = 5$ perform *Operation 1* one time to generate it. For Array c_2, you can choose $i = 6$ and $j = 7$ perform *Operation 1* one time to generate it. For Array c_3,you can choose $i = 4$ and $j = 5$ perform *Operation 2* one time to generate it.

In the second test case, the secret array b is $[20, 20, 20, 20, 20, 20, 20]$. You can also find that array c_1 and array c_2 are generated by using *Operation 1*. Array c_3 is generated by using *Operation 2*.

In the third test case, the secret array b is $[20, 20, 20, 20, 20, 20, 20, 20, 20]$. You can also find that array c_1 and array c_2 are generated by using *Operation 1*. Array c_3 is generated by using *Operation 2*.

E. Count Seconds

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

Cirno has a DAG (Directed Acyclic Graph) with n nodes and m edges. The graph has exactly one node that has no out edges. The i-th node has an integer a_i on it.

Every second the following happens:

- Let S be the set of nodes x that have $a_x > 0$.
- For all $x \in S$, 1 is subtracted from a_x, and then for each node y, such that there is an edge from x to y, 1 is added to a_y.

Find the first moment of time when all a_i become 0. Since the answer can be very large, output it modulo $998\,244\,353$.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases. Description of test cases follows.

The first line of each test case contains two integers n, m ($1 \le n, m \le 1000$) — the number of vertices and edges in the graph.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) — the integer on vertices.

Each line of the following m lines contains two integers x, y ($1 \le x, y \le n$), represent a directed edge from x to y. It is guaranteed that the graph is a DAG with no multi-edges, and there is exactly one node that has no out edges.

It is guaranteed that both sum of n and sum of m over all test cases are less than or equal to $10\,000$.

Output

For each test case, print an integer in a separate line — the first moment of time when all a_i become 0, modulo $998\,244\,353$.

Example

input

```
5
3 2
1 1 1
1 2
2 3
5 5
1 0 0 0 0
1 2
2 3
3 4
4 5
1 5
10 11
998244353 0 0 0 998244353 0 0 0 0 0
1 2
2 3
3 4
4 5
```

```
5 6
6 7
7 8
8 9
9 10
1 3
7 9
5 6
1293 1145 9961 9961 1919
1 2
2 3
3 4
5 4
1 4
2 4
6 9
10 10 10 10 10 10
1 2
1 3
2 3
4 3
6 3
3 5
6 5
6 1
6 2
```

output

```
3
5
4
28010
110
```

Note

In the first test case:

- At time 0, the values of the nodes are $[1, 1, 1]$.
- At time 1, the values of the nodes are $[0, 1, 1]$.
- At time 2, the values of the nodes are $[0, 0, 1]$.
- At time 3, the values of the nodes are $[0, 0, 0]$.

So the answer is 3.

In the second test case:

- At time 0, the values of the nodes are $[1, 0, 0, 0, 0]$.
- At time 1, the values of the nodes are $[0, 1, 0, 0, 1]$.
- At time 2, the values of the nodes are $[0, 0, 1, 0, 0]$.
- At time 3, the values of the nodes are $[0, 0, 0, 1, 0]$.
- At time 4, the values of the nodes are $[0, 0, 0, 0, 1]$.
- At time 5, the values of the nodes are $[0, 0, 0, 0, 0]$.

So the answer is 5.

In the third test case:

The first moment of time when all a_i become 0 is $6 \cdot 998244353 + 4$.

F. Colouring Game

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

Alice and Bob are playing a game. There are n cells in a row. Initially each cell is either red or blue. Alice goes first.

On each turn, Alice chooses two neighbouring cells which contain at least one red cell, and paints that two cells white. Then, Bob chooses two neighbouring cells which contain at least one blue cell, and paints that two cells white. The player who cannot make a move loses.

Find the winner if both Alice and Bob play optimally.

Note that a chosen cell can be white, as long as the other cell satisfies the constraints.

Input

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

For each test case, the first line contains an integer n ($2 \leq n \leq 5 \cdot 10^5$) — the number of cells.

The second line contains a string s of length n — the initial state of the cells. The i-th cell is red if $s_i =$ R, blue if $s_i =$ B.

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

Output
For each test case, output the name of the winner on a separate line.

Example

input
8 3 BRB 5 RRBBB 6 RBRBRB 8 BBRRBRRB 6 BRRBRB 12 RBRBRBRBRRBB 12 RBRBRBRBBBRR 4 RBBR
output
Bob Bob Alice Alice Alice Alice Bob Bob

Note
In the notes, the cell numbers increase from left to right.

In the first testcase for Alice, she has two choices: paint the first and the second cells, or paint the second and the third cells. No matter what choice Alice makes, there will be exactly one blue cell after Alice's move. Bob just needs to paint the blue cell and its neighbour, then every cell will be white and Alice can't make a move. So Bob is the winner.

In the second testcase no matter what Alice chooses, Bob can choose to paint the fourth and fifth cells in 2 turns.

In the third testcase at first, Alice paints the third and the fourth cells. It doesn't matter if Bob paints the first and the second cells or the fifth and sixth cells, as Alice can paint the other two cells.

In the fourth testcase at first, Alice paints the second and the third cells. If Bob paints the fifth and the sixth cells or the fourth and the fifth cells, then Alice paints the seventh and the eighth cells. If Bob paints the seventh and the eighth cells, then Alice paints the fifth and the sixth cells.

In the fifth Alice chooses the middle two cells at first, then Bob obviously has only two options, whichever variant he chooses, Alice can choose the other one and win.

In the eighth no matter what Alice chooses, Bob can choose the other two symmetrical cells.

G. Mio and Lucky Array

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

Mio has an array a consisting of n integers, and an array b consisting of m integers.

Mio can do the following operation to a:

- Choose an integer i ($1 \leq i \leq n$) that has **not** been chosen before, then add 1 to a_i, subtract 2 from a_{i+1}, add 3 to a_{i+2} an so on. Formally, the operation is to add $(-1)^{j-i} \cdot (j-i+1)$ to a_j for $i \leq j \leq n$.

Mio wants to transform a so that it will contain b as a **subarray**. Could you answer her question, and provide a sequence of operations to do so, if it is possible?

An array b is a **subarray** of an array a if b can be obtained from a by deletion of several (possibly, zero or all) elements from the beginning and several (possibly, zero or all) elements from the end.

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

The first line of each test case contains one integer n ($2 \leq n \leq 2 \cdot 10^5$) — the number of elements in a.

The second line of the test case contains n integers $a_1, a_2, \cdots, a_n$ ($-10^5 \leq a_i \leq 10^5$), where a_i is the i-th element of a.

The third line of the test case contains one integer m ($2 \le m \le n$) — the number of elements in b.

The fourth line of the test case contains m integers $b_1, b_2, \cdots, b_m$ ($-10^{12} \le b_i \le 10^{12}$), where b_i is the i-th element of b.

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

Output

If it is impossible to transform a so that it contains b as a subarray, output -1.

Otherwise, the first line of output should contain an integer k ($0 \le k \le n$), the number of operations to be done.

The second line should contain k distinct integers, representing the operations done in order.

If there are multiple solutions, you can output any.

Notice that you do **not** need to minimize the number of operations.

Example

input
5 5 1 2 3 4 5 5 2 0 6 0 10 5 1 2 3 4 5 3 3 5 3 8 -3 2 -3 -4 4 0 1 -2 4 10 -6 7 -7 5 1 2 3 4 5 4 1 10 1 1 5 0 0 0 0 0 2 10 12
output
1 1 1 4 5 1 3 4 6 8 -1 -1

Note

In the first test case, the sequence $a = [1, 2, 3, 4, 5]$. One of the possible solutions is doing one operation at $i = 1$ (add 1 to a_1, subtract 2 from a_2, add 3 to a_3, subtract 4 from a_4, add 5 to a_5). Then array a is transformed to $a = [2, 0, 6, 0, 10]$, which contains $b = [2, 0, 6, 0, 10]$ as a subarray.

In the second test case, the sequence $a = [1, 2, 3, 4, 5]$. One of the possible solutions is doing one operation at $i = 4$ (add 1 to a_4, subtract 2 from a_5). Then array a is transformed to $a = [1, 2, 3, 5, 3]$, which contains $b = [3, 5, 3]$ as a subarray.

In the third test case, the sequence $a = [-3, 2, -3, -4, 4, 0, 1, -2]$. One of the possible solutions is the following.

- Choose an integer $i = 8$ to do the operation. Then array a is transformed to $a = [-3, 2, -3, -4, 4, 0, 1, -1]$.
- Choose an integer $i = 6$ to do the operation. Then array a is transformed to $a = [-3, 2, -3, -4, 4, 1, -1, 2]$.
- Choose an integer $i = 4$ to do the operation. Then array a is transformed to $a = [-3, 2, -3, -3, 2, 4, -5, 7]$.
- Choose an integer $i = 3$ to do the operation. Then array a is transformed to $a = [-3, 2, -2, -5, 5, 0, 0, 1]$.
- Choose an integer $i = 1$ to do the operation. Then array a is transformed to $a = [-2, 0, 1, -9, 10, -6, 7, -7]$.

The resulting a is $[-2, 0, 1, -9, 10, -6, 7, -7]$, which contains $b = [10, -6, 7, -7]$ as a subarray.

In the fourth test case, it is impossible to transform a so that it contains b as a subarray.

In the fifth test case, it is impossible to transform a so that it contains b as a subarray.

H1. Game of AI (easy version)

time limit per test: 2.5 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

This is the easy version of this problem. The difference between easy and hard versions is the constraint on k and the

time limit. Also, in this version of the problem, you only need to calculate the answer when $n = k$. You can make hacks only if both versions of the problem are solved.

Cirno is playing a war simulator game with n towers (numbered from 1 to n) and n bots (numbered from 1 to n). The i-th tower is initially occupied by the i-th bot for $1 \leq i \leq n$.

Before the game, Cirno first chooses a permutation $p = [p_1, p_2, \ldots, p_n]$ of length n (A permutation of length n is an array of length n where each integer between 1 and n appears exactly once). After that, she can choose a sequence $a = [a_1, a_2, \ldots, a_n]$ ($1 \leq a_i \leq n$ and $a_i \neq i$ for all $1 \leq i \leq n$).

The game has n rounds of attacks. In the i-th round, if the p_i-th bot is still in the game, it will begin its attack, and as the result the a_{p_i}-th tower becomes occupied by the p_i-th bot; the bot that previously occupied the a_{p_i}-th tower will no longer occupy it. If the p_i-th bot is not in the game, nothing will happen in this round.

After each round, if a bot doesn't occupy any towers, it will be eliminated and leave the game. Please note that no tower can be occupied by more than one bot, but one bot can occupy more than one tower during the game.

At the end of the game, Cirno will record the result as a sequence $b = [b_1, b_2, \ldots, b_n]$, where b_i is the number of the bot that occupies the i-th tower at the end of the game.

However, as a mathematics master, she wants you to solve the following counting problem instead of playing games:

Count the number of different pairs of sequences a and b that we can get from all possible choices of sequence a and permutation p .

Since this number may be large, output it modulo M.

Input

The only line contains two positive integers k and M ($1 \leq k \leq 5000$, $2 \leq M \leq 10^9$). It is guaranteed that 2^{18} is a divisor of $M - 1$ and M is a prime number.

You need to calculate the answer for $n = k$.

Output

Output a single integer — the number of different pairs of sequences for $n = k$ modulo M.

Examples

input
1 998244353
output
0

input
2 998244353
output
2

input
3 998244353
output
24

input
8 998244353
output
123391016

Note

For $n = 1$, no valid sequence a exists. We regard the answer as 0.

For $n = 2$, there is only one possible array a: $[2, 1]$.

- For array a is $[2, 1]$ and permutation p is $[1, 2]$, the sequence b will be $[1, 1]$ after all rounds have finished. The details for each rounds:
  - In the first round, the first bot will begin its attack and successfully capture the tower 2. After this round, the second bot will be eliminated and leave the game as all of its towers are occupied by other bots.
  - In the second round, the second bot is not in the game.
- For array a is $[2, 1]$ and permutation p is $[2, 1]$, the sequence b will be $[2, 2]$ after all rounds have finished. The details for each rounds:

- In the first round, the second bot will begin its attack and successfully capture the tower 1. After this round, the first bot will be eliminated and leave the game as all of its towers are occupied by other bots.
- In the second round, the first bot is not in the game.

So the number of different pairs of sequences (a, b) is 2 ($[2, 1]$, $[1, 1]$ and $[2, 1]$, $[2, 2]$) for $n = 2$.

H2. Game of AI (hard version)

time limit per test: 12 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

This is the hard version of this problem. The difference between easy and hard versions is the constraint on k and the time limit. Notice that you need to calculate the answer for all positive integers $n \in [1, k]$ in this version. You can make hacks only if both versions of the problem are solved.

Cirno is playing a war simulator game with n towers (numbered from 1 to n) and n bots (numbered from 1 to n). The i-th tower is initially occupied by the i-th bot for $1 \leq i \leq n$.

Before the game, Cirno first chooses a permutation $p = [p_1, p_2, \ldots, p_n]$ of length n (A permutation of length n is an array of length n where each integer between 1 and n appears exactly once). After that, she can choose a sequence $a = [a_1, a_2, \ldots, a_n]$ ($1 \leq a_i \leq n$ and $a_i \neq i$ for all $1 \leq i \leq n$).

The game has n rounds of attacks. In the i-th round, if the p_i-th bot is still in the game, it will begin its attack, and as the result the a_{p_i}-th tower becomes occupied by the p_i-th bot; the bot that previously occupied the a_{p_i}-th tower will no longer occupy it. If the p_i-th bot is not in the game, nothing will happen in this round.

After each round, if a bot doesn't occupy any towers, it will be eliminated and leave the game. Please note that no tower can be occupied by more than one bot, but one bot can occupy more than one tower during the game.

At the end of the game, Cirno will record the result as a sequence $b = [b_1, b_2, \ldots, b_n]$, where b_i is the number of the bot that occupies the i-th tower at the end of the game.

However, as a mathematics master, she wants you to solve the following counting problem instead of playing games:

Count the number of different pairs of sequences a, b from all possible choices of sequence a and permutation p.

Calculate the answers for all n such that $1 \leq n \leq k$. Since these numbers may be large, output them modulo M.

Input

The only line contains two positive integers k and M ($1 \leq k \leq 10^5$, $2 \leq M \leq 10^9$). It is guaranteed that 2^{18} is a divisor of $M - 1$ and M is a prime number.

Output

Output k lines, where the i-th line contains a non-negative integer, which is the answer for $n = i$ modulo M.

Example

input
8 998244353
output
0 2 24 360 6800 153150 4057452 123391016

Note

For $n = 1$, no valid sequence a exists. We regard the answer as 0.

For $n = 2$, there is only one possible array a: $[2, 1]$.

- For array a is $[2, 1]$ and permutation p is $[1, 2]$, the sequence b will be $[1, 1]$ after all rounds have finished. The details for each rounds:
  - In the first round, the first bot will begin its attack and successfully capture the tower 2. After this round, the second bot will be eliminated and leave the game as all of its towers are occupied by other bots.
  - In the second round, the second bot is not in the game.
- For array a is $[2, 1]$ and permutation p is $[2, 1]$, the sequence b will be $[2, 2]$ after all rounds have finished. The details for each rounds:
  - In the first round, the second bot will begin its attack and successfully capture the tower 1. After this round, the first bot will

be eliminated and leave the game as all of its towers are occupied by other bots.
- In the second round, the first bot is not in the game.

So the number of different pairs of sequences (a, b) is 2 ($[2, 1]$, $[1, 1]$ and $[2, 1]$, $[2, 2]$) for $n = 2$.
