A. Shohag Loves Mod

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

Shohag has an integer n. Please help him find an **increasing** integer sequence $1 \le a_1 < a_2 < \ldots < a_n \le 100$ such that $a_i \bmod i \neq a_j \bmod j$ * is satisfied over all pairs $1 \le i < j \le n$.

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

---

* $a \bmod b$ denotes the remainder of a after division by b. For example, $7 \bmod 3 = 1, 8 \bmod 4 = 0$ and $69 \bmod 10 = 9$.

Input

The first line contains a single integer t ($1 \le t \le 50$) — the number of test cases.

The first and only line of each test case contains an integer n ($2 \le n \le 50$).

Output

For each test case, print n integers — the integer sequence that satisfies the conditions mentioned in the statement. If there are multiple such sequences, output any.

Standard Input	Standard Output
2 3 6	2 7 8 2 3 32 35 69 95

Note

In the first test case, the sequence is increasing, values are from 1 to 100 and each pair of indices satisfies the condition mentioned in the statement:

- For pair $(1, 2)$, $a_1 \bmod 1 = 2 \bmod 1 = 0$, and $a_2 \bmod 2 = 7 \bmod 2 = 1$. So they are different.
- For pair $(1, 3)$, $a_1 \bmod 1 = 2 \bmod 1 = 0$, and $a_3 \bmod 3 = 8 \bmod 3 = 2$. So they are different.
- For pair $(2, 3)$, $a_2 \bmod 2 = 7 \bmod 2 = 1$, and $a_3 \bmod 3 = 8 \bmod 3 = 2$. So they are different.

Note that you do not necessarily have to print the exact same sequence, you can print any other sequence as long as it satisfies the necessary conditions.

,

B. Shohag Loves Strings

Input file: standard input
Output file: standard output
Time limit: 1 second
Memory limit: 256 megabytes

For a string p, let $f(p)$ be the number of distinct non-empty substrings* of p.

Shohag has a string s. Help him find a non-empty string p such that p is a substring of s and $f(p)$ is even or state that no such string exists.

---

*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.

Input

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

The first and only line of each test case contains a string s ($1 \leq |s| \leq 10^5$) consisting of lowercase English letters.

It is guaranteed that the sum of the length of s over all test cases doesn't exceed $3 \cdot 10^5$.

Output

For each test case, print a non-empty string that satisfies the conditions mentioned in the statement, or -1 if no such string exists. If there are multiple solutions, output any.

Standard Input	Standard Output
5 dcabaac a youknowwho codeforces bangladesh	abaa -1 youknowwho eforce bang

Note

In the first test case, we can set $p = \texttt{abaa}$ because it is a substring of s and the distinct non-empty substrings of p are a, b, aa, ab, ba, aba, baa and abaa, so it has a total of 8 distinct substrings which is even.

In the second test case, we can only set $p = \texttt{a}$ but it has one distinct non-empty substring but this number is odd, so not valid.

In the third test case, the whole string contains 52 distinct non-empty substrings, so the string itself is a valid solution.

,

C1. Shohag Loves XOR (Easy Version)

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

This is the easy version of the problem. The differences between the two versions are highlighted in bold. You can only make hacks if both versions of the problem are solved.

Shohag has two integers x and m. Help him count the number of integers $1 \le y \le m$ such that $\mathbf{x \neq y}$ and $x \oplus y$ is **a divisor*** **of** either x, y, or both. Here $\oplus$ is the bitwise XOR operator.

*The number b is a divisor of the number a if there exists an integer c such that $a = b \cdot c$.

Input

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

The first and only line of each test case contains two space-separated integers x and m ($1 \le x \le 10^6$, $1 \le m \le 10^{18}$).

It is guaranteed that the sum of x over all test cases does not exceed 10^7.

Output

For each test case, print a single integer — the number of suitable y.

Standard Input	Standard Output
5 6 9 5 7 2 3 6 4 4 1	3 2 1 1 0

Note

In the first test case, for $x = 6$, there are 3 valid values for y among the integers from 1 to $m = 9$, and they are 4, 5, and 7.

- $y = 4$ is valid because $x \oplus y = 6 \oplus 4 = 2$ and 2 is a divisor of both $x = 6$ and $y = 4$.
- $y = 5$ is valid because $x \oplus y = 6 \oplus 5 = 3$ and 3 is a divisor of $x = 6$.
- $y = 7$ is valid because $x \oplus y = 6 \oplus 7 = 1$ and 1 is a divisor of both $x = 6$ and $y = 7$.

In the second test case, for $x = 5$, there are 2 valid values for y among the integers from 1 to $m = 7$, and they are 4 and 6.

- $y = 4$ is valid because $x \oplus y = 5 \oplus 4 = 1$ and 1 is a divisor of both $x = 5$ and $y = 4$.
- $y = 6$ is valid because $x \oplus y = 5 \oplus 6 = 3$ and 3 is a divisor of $y = 6$.

,

C2. Shohag Loves XOR (Hard Version)

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

This is the hard version of the problem. The differences between the two versions are highlighted in bold. You can only make hacks if both versions of the problem are solved.

Shohag has two integers x and m. Help him count the number of integers $1 \le y \le m$ such that $x \oplus y$ is **divisible*** **by** either x, y, or both. Here $\oplus$ is the bitwise XOR operator.

*The number a is divisible by the number b if there exists an integer c such that $a = b \cdot c$.

Input

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

The first and only line of each test case contains two space-separated integers x and m ($1 \le x \le 10^6$, $1 \le m \le 10^{18}$).

It is guaranteed that the sum of x over all test cases does not exceed 10^7.

Output

For each test case, print a single integer — the number of suitable y.

Standard Input	Standard Output
5 7 10 2 3 6 4 1 6 4 1	3 2 2 6 1

Note

In the first test case, for $x = 7$, there are 3 valid values for y among the integers from 1 to $m = 10$, and they are 1, 7, and 9.

- $y = 1$ is valid because $x \oplus y = 7 \oplus 1 = 6$ and 6 is divisible by $y = 1$.
- $y = 7$ is valid because $x \oplus y = 7 \oplus 7 = 0$ and 0 is divisible by both $x = 7$ and $y = 7$.
- $y = 9$ is valid because $x \oplus y = 7 \oplus 9 = 14$ and 14 is divisible by $x = 7$.

,

D. Shohag Loves GCD

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

Shohag has an integer n and a set S of m unique integers. Help him find the lexicographically largest* integer array $a_1, a_2, \ldots, a_n$ such that $a_i \in S$ for each $1 \leq i \leq n$ and $a_{\gcd(i,j)} \neq \gcd(a_i, a_j)^\dagger$ is satisfied over all pairs $1 \leq i < j \leq n$, or state that no such array exists.

*An array a is lexicographically larger than an array b of the same length if $a \neq b$, and in the first position where a and b differ, the array a has a larger element than the corresponding element in b.

$^\dagger \gcd(x, y)$ denotes the greatest common divisor (GCD) of integers x and y.

Input

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

The first line of each test case contains two integers n and m ($1 \leq m \leq n \leq 10^5$).

The second line contains m unique integers in increasing order, representing the elements of the set S ($1 \leq x \leq n$ for each $x \in S$).

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, if there is no solution print -1, otherwise print n integers — the lexicographically largest integer array that satisfies the conditions.

Standard Input	Standard Output
3 6 3 3 4 6 1 1 1 2 1 2	6 4 4 3 4 3 1 -1

Note

In the first test case, every element in the array belongs to the given set $S = \{3, 4, 6\}$, and all pairs of indices of the array satisfy the necessary conditions. In particular, for pair $(2, 3)$, $a_{\gcd(2,3)} = a_1 = 6$ and $\gcd(a_2, a_3) = \gcd(4, 4) = 4$, so they are not equal. There are other arrays that satisfy the conditions as well but this one is the lexicographically largest among them.

In the third test case, there is no solution possible because we are only allowed to use $a = [2, 2]$ but for this array, for pair $(1, 2)$, $a_{\gcd(1,2)} = a_1 = 2$ and $\gcd(a_1, a_2) = \gcd(2, 2) = 2$, so they are equal which is not allowed!

,

E. Shohag Loves Inversions

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 256 megabytes

Shohag has an array a of integers. Initially $a = [0, 1]$. He can repeatedly perform the following operation any number of times:

- Let k be the number of inversions* in the current array a.
- Insert k at any position in a, including the beginning or the end.

For example, if $a = [4, 6, 2, 4]$, then the number of inversions is $k = 3$. So Shohag can obtain the following arrays after the operation: $[\mathbf{3}, 4, 6, 2, 4]$, $[4, \mathbf{3}, 6, 2, 4]$, $[4, 6, \mathbf{3}, 2, 4]$, $[4, 6, 2, \mathbf{3}, 4]$, and $[4, 6, 2, 4, \mathbf{3}]$.

Given an integer n, help Shohag count, modulo $998\,244\,353$, the number of distinct arrays of length n that can be obtained after performing the operations.

*The number of inversions in an array a is the number of pairs of indices (i, j) such that $i < j$ and $a_i > a_j$.

Input

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

The first and only line of each test case contains an integer n ($2 \le n \le 10^6$).

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

Output

For each test case, output an integer — the number of possible arrays modulo $998\,244\,353$.

Standard Input	Standard Output
4 4 2 7 69	5 1 682 325188814

Note

In the first test case, the following 5 arrays can be obtained (the inserted inversion count is shown in bold):

- $[0, 1] \to [0, \mathbf{0}, 1] \to [0, 0, 1, \mathbf{0}]$,
- $[0, 1] \to [0, \mathbf{0}, 1] \to [0, 0, \mathbf{0}, 1]$,
- $[0, 1] \to [0, 1, \mathbf{0}] \to [0, 1, 0, \mathbf{1}]$,
- $[0, 1] \to [0, 1, \mathbf{0}] \to [0, 1, \mathbf{1}, 0]$,
- $[0, 1] \to [0, 1, \mathbf{0}] \to [\mathbf{1}, 0, 1, 0]$.

,

F1. Shohag Loves Counting (Easy Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     4 seconds
Memory limit:   512 megabytes
```

This is the easy version of the problem. The only differences between the two versions of this problem are the constraints on t, m, and the sum of m. You can only make hacks if both versions of the problem are solved.

For an integer array a of length n, define $f(k)$ as the greatest common divisor (GCD) of the maximum values of all subarrays* of length k. For example, if the array is $[2, 1, 4, 6, 2]$, then $f(3) = \gcd(\max([2, 1, 4]), \max([1, 4, 6]), \max([4, 6, 2])) = \gcd(4, 6, 6) = 2$.

An array is good if $f(i) \neq f(j)$ is satisfied over all pairs $1 \leq i < j \leq n$.

Shohag has an integer m. Help him count the number, modulo $998\,244\,353$, of non-empty good arrays of arbitrary length such that each element of the array is an integer from 1 to m.

---

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

Input

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

The first and only line of each test case contains an integer m ($1 \leq m \leq 10^5$).

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 — the number of valid arrays modulo $998\,244\,353$.

Standard Input	Standard Output
3 2 5 9	4 29 165

Note

In the first test case, the valid arrays are $[1]$, $[1, 2]$, $[2]$, and $[2, 1]$.

In the second test case, there are a total of 29 valid arrays. In particular, the array $[2, 1, 4]$ with length $n = 3$ is valid because all elements are from 1 to $m = 5$ and $f(1)$, $f(2)$ and $f(n = 3)$ all are distinct:

- $f(1) = \gcd(\max([2]), \max([1]), \max([4])) = \gcd(2, 1, 4) = 1$.
- $f(2) = \gcd(\max([2, 1]), \max([1, 4])) = \gcd(2, 4) = 2$.
- $f(3) = \gcd(\max([2, 1, 4])) = \gcd(4) = 4$.

,

F2. Shohag Loves Counting (Hard Version)

Input file: standard input
Output file: standard output
Time limit: 4 seconds
Memory limit: 512 megabytes

This is the hard version of the problem. The only differences between the two versions of this problem are the constraints on t, m, and the sum of m. You can only make hacks if both versions of the problem are solved.

For an integer array a of length n, define $f(k)$ as the greatest common divisor (GCD) of the maximum values of all subarrays* of length k. For example, if the array is $[2, 1, 4, 6, 2]$, then $f(3) = \gcd(\max([2, 1, 4]), \max([1, 4, 6]), \max([4, 6, 2])) = \gcd(4, 6, 6) = 2$.

An array is good if $f(i) \neq f(j)$ is satisfied over all pairs $1 \leq i < j \leq n$.

Shohag has an integer m. Help him count the number, modulo $998\,244\,353$, of non-empty good arrays of arbitrary length such that each element of the array is an integer from 1 to m.

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

Input

The first line contains a single integer t ($1 \leq t \leq 3 \cdot 10^5$) — the number of test cases.

The first and only line of each test case contains an integer m ($1 \leq m \leq 10^6$).

Note that there is no limit on the sum of m over all test cases.

Output

For each test case, output an integer — the number of valid arrays modulo $998\,244\,353$.

Standard Input	Standard Output
3 2 5 9	4 29 165

Note

In the first test case, the valid arrays are $[1]$, $[1, 2]$, $[2]$, and $[2, 1]$.

In the second test case, there are a total of 29 valid arrays. In particular, the array $[2, 1, 4]$ with length $n = 3$ is valid because all elements are from 1 to $m = 5$ and $f(1)$, $f(2)$ and $f(n = 3)$ all are distinct:

- $f(1) = \gcd(\max([2]), \max([1]), \max([4])) = \gcd(2, 1, 4) = 1$.
- $f(2) = \gcd(\max([2, 1]), \max([1, 4])) = \gcd(2, 4) = 2$.
- $f(3) = \gcd(\max([2, 1, 4])) = \gcd(4) = 4$.

,

G. Shohag Loves Pebae

```
Input file:     standard input
Output file:    standard output
Time limit:     5 seconds
Memory limit:   768 megabytes
```

Shohag has a tree with n nodes.

Pebae has an integer m. She wants to assign each node a value — an integer from 1 to m. So she asks Shohag to count the number, modulo $998\,244\,353$, of assignments such that following conditions are satisfied:

- For each pair $1 \leq u < v \leq n$, the least common multiple (LCM) of the values of the nodes in the unique simple path from u to v is **not** divisible by the number of nodes in the path.
- The greatest common divisor (GCD) of the values of all nodes from 1 to n is 1.

But this problem is too hard for Shohag to solve. As Shohag loves Pebae, he has to solve the problem. Please save Shohag!

Input

The first line contains two space-separated integers n and m ($2 \leq n \leq 10^6$, $1 \leq m \leq 10^9$).

Each of the next $n - 1$ lines contains two integers u and v ($1 \leq u, v \leq n$) indicating there is an edge between vertices u and v. It is guaranteed that the given edges form a tree.

Output

Print a single integer — the number of valid ways to assign each vertex a value, modulo $998\,244\,353$.

Standard Input	Standard Output
6 6 1 2 2 3 3 4 4 5 3 6	2
2 5 1 2	7
12 69 3 5 1 4 2 3 4 5 5 6 8 9 7 3 4 8 9 10 1 11 12 1	444144548

Note

In the first test case, the valid assignments are $[1, 1, 1, 1, 1, 1]$ and $[1, 1, 1, 1, 1, 5]$.

In the second test case, the valid assignments are $[1, 1]$, $[1, 3]$, $[1, 5]$, $[3, 1]$, $[3, 5]$, $[5, 1]$ and $[5, 3]$.

,

H1. Cool Swap Walk (Easy Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

This is the easy version of the problem. The only difference is the maximum number of operations you can perform. You can only make hacks if both versions are solved.

You are given an array a of size n.

A *cool swap walk* is the following process:

- In an $n \times n$ grid, we note the cells in row i and column j as (i, j). You need to walk from $(1, 1)$ to (n, n), taking only steps to the right or down.
- Formally, if you are in (x, y) currently, you can step to either $(x + 1, y)$ or $(x, y + 1)$, but you can not step beyond the boundaries of the grid.
- When you step in (i, j), you **must** swap a_i and a_j when $i \neq j$.

You can perform at most $2n + 4$ *cool swap walks*. Sort the array $a_1, a_2, \ldots, a_n$ in non-decreasing order. We can show that it's always possible to do so.

Input

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

The first line of each test case contains an integer n ($2 \leq n \leq 500$) — the size of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$) — the elements of the array.

It is guaranteed that the sum of n^2 over all test cases does not exceed $2.5 \cdot 10^5$.

Output

For each test case, your output should consist of several lines:

- The first line contains an integer k ($0 \leq k \leq 2n + 4$), representing the number of *cool swap walks* you perform.
- Each of the next k lines contains a string s of length $2n - 2$ consisting only of R and D, representing the path (letters are case sensitive). For all $1 \leq i \leq 2n - 2$, if $s_i =$ R, you walk right in the i-th step, otherwise you walk down in the i-th step.

Standard Input	Standard Output
3 2 1 2 3 2 1 3 4 3 2 3 4	0 2 RRDD DRDR 3 RRDRDD DRDDRR DDRRRD

Note

In the first test case, the array a is already non-decreasing, so you don't need to perform any walk.

In the second test case, $a = [2, 1, 3]$ initially.

In the first walk:

- In the 1-st step, you step right to $(1, 2)$. Then, $a = [1, 2, 3]$. Note that although the array a is already non-decreasing, you can **not** stop until you reach (n, n).
- In the 2-nd step, you step right to $(1, 3)$. Then, $a = [3, 2, 1]$.
- In the 3-rd step, you step down to $(2, 3)$. Then, $a = [3, 1, 2]$.
- In the 4-th step, you step down to $(3, 3)$. Then, $a = [3, 1, 2]$.

In the second walk:

- In the 1-st step, you step down to $(2, 1)$. Then, $a = [1, 3, 2]$.
- In the 2-nd step, you step right to $(2, 2)$. Then, $a = [1, 3, 2]$.
- In the 3-rd step, you step down to $(3, 2)$. Then, $a = [1, 2, 3]$.
- In the 4-th step, you step down to $(3, 3)$. Then, $a = [1, 2, 3]$.

After the two *cool swap walks* above, we get $a = [1, 2, 3]$, which is non-decreasing.

,

H2. Cool Swap Walk (Hard Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

This is the hard version of the problem. The only difference is the maximum number of operations you can perform. You can only make hacks if both versions are solved.

You are given an array a of size n.

A *cool swap walk* is the following process:

- In an $n \times n$ grid, we note the cells in row i and column j as (i, j). You need to walk from $(1, 1)$ to (n, n), taking only steps to the right or down.
- Formally, if you are in (x, y) currently, you can step to either $(x + 1, y)$ or $(x, y + 1)$, but you can not step beyond the boundaries of the grid.
- When you step in (i, j), you **must** swap a_i and a_j when $i \neq j$.

You can perform at most $n + 4$ *cool swap walks*. Sort the array $a_1, a_2, \ldots, a_n$ in non-decreasing order. We can show that it's always possible to do so.

Input

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

The first line of each test case contains an integer n ($2 \leq n \leq 500$) — the size of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$) — the elements of the array.

It is guaranteed that the sum of n^2 over all test cases does not exceed $2.5 \cdot 10^5$.

Output

For each test case, your output should consist of several lines:

- The first line contains an integer k ($0 \leq k \leq n + 4$), representing the number of *cool swap walks* you perform.
- Each of the next k lines contains a string s of length $2n - 2$ consisting only of R and D, representing the path (letters are case sensitive). For all $1 \leq i \leq 2n - 2$, if $s_i =$ R, you walk right in the i-th step, otherwise you walk down in the i-th step.

Standard Input	Standard Output
3 2 1 2 3 2 1 3 4 3 2 3 4	0 2 RRDD DRDR 3 RRDRDD DRDDRR DDRRRD

Note

In the first test case, the array a is already non-decreasing, so you don't need to perform any walk.

In the second test case, $a = [2, 1, 3]$ initially.

In the first walk:

- In the 1-st step, you step right to $(1, 2)$. Then, $a = [1, 2, 3]$. Note that although the array a is already non-decreasing, you can **not** stop until you reach (n, n).
- In the 2-nd step, you step right to $(1, 3)$. Then, $a = [3, 2, 1]$.
- In the 3-rd step, you step down to $(2, 3)$. Then, $a = [3, 1, 2]$.
- In the 4-th step, you step down to $(3, 3)$. Then, $a = [3, 1, 2]$.

In the second walk:

- In the 1-st step, you step down to $(2, 1)$. Then, $a = [1, 3, 2]$.
- In the 2-nd step, you step right to $(2, 2)$. Then, $a = [1, 3, 2]$.
- In the 3-rd step, you step down to $(3, 2)$. Then, $a = [1, 2, 3]$.
- In the 4-th step, you step down to $(3, 3)$. Then, $a = [1, 2, 3]$.

After the two *cool swap walks* above, we get $a = [1, 2, 3]$, which is non-decreasing.