A. Kevin and Combination Lock

1 second, 256 megabytes

Kevin is trapped in Lakeside Village by Grace. At the exit of the village, there is a combination lock that can only be unlocked if Kevin solves it.

The combination lock starts with an integer x. Kevin can perform one of the following two operations zero or more times:

1. If $x \neq 33$, he can select two consecutive digits 3 from x and remove them simultaneously. For example, if $x = 13\,323$, he can remove the second and third 3, changing x to 123.
2. If $x \geq 33$, he can change x to $x - 33$. For example, if $x = 99$, he can choose this operation to change x to $99 - 33 = 66$.

When the value of x on the combination lock becomes 0, Kevin can unlock the lock and escape from Lakeside Village. Please determine whether it is possible for Kevin to unlock the combination lock and escape.

Input

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

The only line of each test case contains a positive integer x ($1 \leq x \leq 10^9$).

Output

For each test case, output "YES" or "NO" (without quotes) in one line, representing whether Kevin can unlock the combination lock and escape. You can output the answer in any case (upper or lower). For example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as positive responses.

input
5 165 6369 666 114514 133333332

output
YES YES NO NO YES

For the first test case,

$165 \xrightarrow{-33} 132 \xrightarrow{-33} 99 \xrightarrow{-33} 66 \xrightarrow{-33} 33 \xrightarrow{-33} 0$.

For the second test case,

$6369 \xrightarrow{-33} 6336 \xrightarrow{\text{remove "33"}} 66 \xrightarrow{-33} 33 \xrightarrow{-33} 0$.

For the third test case, it can be proven that, regardless of the operations performed, 666 cannot be transformed into 0.

B. Kevin and Permutation

1 second, 256 megabytes

Kevin is a master of permutation-related problems. You are taking a walk with Kevin in Darkwoods, and during your leisure time, he wants to ask you the following question.

Given two positive integers n and k, construct a permutation* p of length n to minimize the sum of the minimum values of all subarrays† of length k. Formally, you need to minimize

$$\sum_{i=1}^{n-k+1} \left(\min_{j=i}^{i+k-1} p_j \right).$$

*A permutation of length n is an array consisting of n distinct integers from 1 to n in arbitrary order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (2 appears twice in the array), and $[1, 3, 4]$ is also not a permutation ($n = 3$ but there is 4 in the array).

†An array a is a subarray of an array b if a can be obtained from b by the deletion of several (possibly, zero or all) elements from the beginning and several (possibly, zero or all) elements from the end. Two subarrays are considered different if the sets of **positions** of the deleted elements are different.

Input

Each test consists of multiple test cases. The first line contains the number of test cases t ($1 \leq t \leq 10^3$).

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

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

Output

For each test case, output n integers on a single line — the permutation p you constructed.

If there are multiple answers, you can print any of them.

input
3 4 2 6 1 8 3

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

In the first test case, with $k = 2$, consider all subarrays of length 2: the minimum value of p_1, p_2 is 1, the minimum value of p_2, p_3 is 1, and the minimum value of p_3, p_4 is 2. The sum $1 + 1 + 2 = 4$ is the smallest among all possible permutations.

In the second test case, all subarrays of length 1 have minimum values of $5, 2, 1, 6, 4, 3$, and the sum $5 + 2 + 1 + 6 + 4 + 3 = 21$ is proven to be the smallest.

C. Kevin and Binary Strings

2 seconds, 256 megabytes

Kevin discovered a binary string s that **starts with 1** in the river at Moonlit River Park and handed it over to you. Your task is to select two non-empty substrings* of s (which can be overlapped) to maximize the XOR value of these two substrings.

The XOR of two binary strings a and b is defined as the result of the $\oplus$ operation applied to the two numbers obtained by interpreting a and b as binary numbers, with the leftmost bit representing the highest value. Here, $\oplus$ denotes the bitwise XOR operation.

The strings you choose may have leading zeros.

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

Input
Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^3$).

The only line of each test case contains a binary string s that **starts with** `1` ($1 \le |s| \le 5000$).

It is guaranteed that the sum of $|s|$ over all test cases doesn't exceed 5000.

Output
For each test case, output four integers l_1, r_1, l_2, r_2 ($1 \le l_1 \le r_1 \le |s|$, $1 \le l_2 \le r_2 \le |s|$) — in the case the two substrings you selected are $s_{l_1} s_{l_1+1} \ldots s_{r_1}$ and $s_{l_2} s_{l_2+1} \ldots s_{r_2}$.

If there are multiple solutions, print any of them.

input
5 111 1000 10111 11101 1100010001101

output
2 2 1 3 1 3 1 4 1 5 1 4 3 4 1 5 1 13 1 11

In the first test case, we can choose s_2 = `1` and $s_1 s_2 s_3$ = `111`, and `1` $\oplus$ `111` = `110`. It can be proven that it is impossible to obtain a larger result. Additionally, $l_1 = 3$, $r_1 = 3$, $l_2 = 1$, $r_2 = 3$ is also a valid solution.

In the second test case, $s_1 s_2 s_3$ = `100`, $s_1 s_2 s_3 s_4$ = `1000`, the result is `100` $\oplus$ `1000` = `1100`, which is the maximum.

D. Kevin and Competition Memories

2 seconds, 256 megabytes

Kevin used to get into Rio's Memories, and in Rio's Memories, a series of contests was once held. Kevin remembers all the participants and all the contest problems from that time, but he has forgotten the specific rounds, the distribution of problems, and the exact rankings.

There are m problems in total, with the i-th problem having a difficulty of b_i. Let each contest consist of k problems, resulting in a total of $\lfloor \frac{m}{k} \rfloor$ contests. This means that you select exactly $\lfloor \frac{m}{k} \rfloor \cdot k$ problems for the contests in any combination you want, with each problem being selected at most once, and the remaining $m \bmod k$ problems are left unused. For example, if $m = 17$ and $k = 3$, you should create exactly 5 contests consisting of 3 problems each, and exactly 2 problems will be left unused.

There are n participants in the contests, with Kevin being the 1-st participant. The i-th participant has a rating of a_i. During the contests, each participant solves all problems with a difficulty not exceeding their rating, meaning the i-th participant solves the j-th problem if and only if $a_i \ge b_j$. In each contest, Kevin's *rank* is one plus the number of participants who solve more problems than he does.

For each $k = 1, 2, \ldots, m$, Kevin wants to know the minimum sum of his ranks across all $\lfloor \frac{m}{k} \rfloor$ contests. In other words, for some value of k, after selecting the problems for each contest, you calculate the rank of Kevin in each contest and sum up these ranks over all $\lfloor \frac{m}{k} \rfloor$ contests. Your goal is to minimize this value.

Note that contests for different values of k are independent. It means that for different values of k, you can select the distribution of problems into the contests independently.

Input
Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 5 \cdot 10^4$).

The first line of each test case contains two integers n and m ($1 \le n, m \le 3 \cdot 10^5$) — the number of participants and the number of problems.

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 rating of each participant.

The third line of each test case contains m integers $b_1, b_2, \ldots, b_m$ ($0 \le b_i \le 10^9$) — the difficulty of each problem.

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

Output
For each test case, output m integers — the minimum sum of Kevin's ranks for each $k = 1, 2, \ldots, m$.

input
4 4 4 4 3 7 5 2 5 4 6 5 5 5 0 4 8 6 1 3 9 2 7 6 7 1 1 4 5 1 4 1 9 1 9 8 1 0 7 6 1 9 1 9 8 1 0 1 1 4 5 1 4

output
7 4 2 3 6 2 1 1 2 7 3 2 1 1 1 1 15 9 5 4 4 4

For the first test case:

When $k = 1$, since each contest only contains one problem, the distribution is in fact unique. For example, in the contest which only includes the third problem (which has a difficulty of 4), all participants except the 2-nd can solve it. Since no one solves strictly more problems than Kevin, his ranking in this contest is 1. Similarly, in all 4 contests, Kevin's rankings are 1, 3, 1, 2, and the sum is 7.

When $k = 2$, one optimal way is to choose the 1-st and the 3-rd problem to form a contest, while the 2-nd and 4-th for another. In the former contest, 4 participants respectively solve 2, 1, 2, 2 problems, so Kevin's ranking is 1; in the latter one, they respectively solve 0, 0, 2, 1, since there are 2 participants (3-rd and 4-th) solve more problems than Kevin, his ranking is 1 + 2 = 3. Thus the answer is 1 + 3 = 4. It can be proven that there's no way to achieve a lower sum.

When $k = 3$, we can simply choose the 1-st, the 3-rd, and the 4-th problem to make a contest, and Kevin has a ranking of 2, which is optimal.

When $k = 4$, since there's only one contest, the distribution is also unique, and Kevin's ranking is 3.

E. Kevin and Bipartite Graph

2 seconds, 256 megabytes

The Arms Factory needs a poster design pattern and finds Kevin for help.

A poster design pattern is a bipartite graph with $2n$ vertices in the left part and m vertices in the right part, where there is an edge between each vertex in the left part and each vertex in the right part, resulting in a total of $2nm$ edges.

Kevin must color each edge with a positive integer in the range $[1, n]$. A poster design pattern is good if there are no monochromatic cycles* in the bipartite graph.

Kevin needs your assistance in constructing a good bipartite graph or informing him if it is impossible.

* A monochromatic cycle refers to a simple cycle in which all the edges are colored with the same color.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$).

The only line of each test case contains two integers n and m ($1 \le n, m \le 10^3$) — the bipartite graph has $2n$ vertices in the left part and m vertices in the right part.

It is guaranteed that both the sum of n and the sum of m over all test cases do not exceed 10^3.

Output

For each test case, if there is no solution, then output `No`.

Otherwise, output `Yes`, and then output $2n$ lines, with each line containing m positive integers. The i-th line's j-th integer represents the color of the edge between the i-th vertex in the left part and the j-th vertex in the right part.

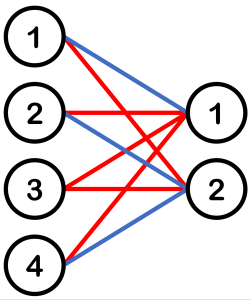
If there are multiple answers, you can print any of them.

You can output each letter in any case (for example, the strings `yEs`, `yes`, `Yes`, and `YES` will be recognized as a positive answer).

input
3 2 2 3 7 5 4

output
YES 1 2 2 1 2 2 2 1 NO YES 1 1 1 1 1 2 2 2 1 2 3 3 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4

For the first test case, the graph is shown as follows:



For the second test case, it can be proven that there is no valid solution.

F. Kevin and Math Class

2 seconds, 1024 megabytes

Kevin is a student from Eversleeping Town, currently attending a math class where the teacher is giving him division exercises.

On the board, there are two rows of positive integers written, each containing n numbers. The first row is $a_1, a_2, \ldots, a_n$, and the second row is $b_1, b_2, \ldots, b_n$.

For each division exercise, Kevin can choose any segment $[l, r]$ and find the smallest value x among $b_l, b_{l+1}, \ldots, b_r$. He will then modify each a_i for $l \le i \le r$ to be the ceiling of a_i divided by x.

Formally, he selects two integers $1 \le l \le r \le n$, sets $x = \min_{l \le i \le r} b_i$, and changes all a_i for $l \le i \le r$ to $\lceil \frac{a_i}{x} \rceil$.

Kevin can leave class and go home when all a_i become 1. He is eager to leave and wants to know the minimum number of division exercises required to achieve this.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$).

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the sequence a and b.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^{18}$) — the first row of integers on the board.

The third line of each test case contains n integers $b_1, b_2, \ldots, b_n$ ($2 \le b_i \le 10^{18}$) — the second row of integers on the board.

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

Output

For each test case, output one integer — the minimum number of division exercises required to leave class.

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

output
2 3 3

For the first test case:

$$[5, 4, 2] \xrightarrow[\min(b_1,b_2)=3]{\text{operate segment } [1,2]} [2, 2, 2] \xrightarrow[\min(b_1,b_2,b_3)=2]{\text{operate segment } [1,3]} [1, 1, 1].$$

For the second test case:

$$[3, 6, 1, 3, 2] \xrightarrow[\min(b_1,b_2,b_3)=3]{\text{operate segment } [1,3]} [1, 2, 1, 3, 2] \xrightarrow[\min(b_2,b_3,b_4)=2]{\text{operate segment } [2,4]} [1, 1, \ldots]$$

.

G. Kevin and Matrices

6 seconds, 256 megabytes

Kevin has been transported to Sacred Heart Hospital, which contains all the $n \times m$ matrices with integer values in the range $[1, v]$.

Now, Kevin wants to befriend some matrices, but he is willing to befriend a matrix a if and only if the following condition is satisfied:

$$\min_{1 \le i \le n} \left(\max_{1 \le j \le m} a_{i,j} \right) \le \max_{1 \le j \le m} \left(\min_{1 \le i \le n} a_{i,j} \right).$$

Please count how many matrices in Sacred Heart Hospital can be friends with Kevin.

Since Kevin is very friendly, there could be many matrices that meet this condition. Therefore, you only need to output the result modulo 998 244 353.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 8 \cdot 10^3$).

The only line of each test case contains three integers n, m, v ($1 \le n, v, n \cdot v \le 10^6$, $1 \le m \le 10^9$).

It is guaranteed that the sum of $n \cdot v$ over all test cases doesn't exceed 10^6.

Output

For each test case, output one integer — the number of matrices that can be friends with Kevin modulo 998 244 353.

input
3 2 2 2 2 3 4 11 45 14

output
14 2824 883799966

In the first test case, besides the matrices $a = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$ and $a = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$, which do not satisfy the condition, the remaining $2^{2 \cdot 2} - 2 = 14$ matrices can all be friends with Kevin.

H. Kevin and Strange Operation

2 seconds, 256 megabytes

Kevin is exploring problems related to binary strings in Chinatown. When he was at a loss, a stranger approached him and introduced a peculiar operation:

- Suppose the current binary string is t, with a length of $|t|$. Choose an integer $1 \le p \le |t|$. For all $1 \le i < p$, **simultaneously** perform the operation $t_i = \max(t_i, t_{i+1})$, and then delete t_p.

For example, suppose the current binary string is `01001`, and you choose $p = 4$. Perform $t_i = \max(t_i, t_{i+1})$ for t_1, t_2, and t_3, transforming the string into `11001`, then delete t_4, resulting in `1101`.

Kevin finds this strange operation quite interesting. Thus, he wants to ask you: Given a binary string s, how many distinct non-empty binary strings can you obtain through any number of operations (possibly zero)?

Since the answer may be very large, you only need to output the result modulo 998 244 353.

Input

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

For each test case, the only line contains a binary string s ($1 \le |s| \le 10^6$).

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

Output

For each test case, print a single integer in the only line of the output — the number of distinct non-empty binary strings you can obtain, modulo 998 244 353.

input
2 11001 000110111001100

output
9 73

In the first test case, all the binary strings you can obtain are: `11001`, `1001`, `1101`, `001`, `101`, `111`, `01`, `11`, and `1`. There are 9 in total.

I1. Kevin and Puzzle (Easy Version)

2 seconds, 512 megabytes

This is the easy version of the problem. The difference between the versions is that in this version, you need to find any one good array. You can hack only if you solved all versions of this problem.

Kevin is visiting the Red Church, and he found a puzzle on the wall.

For an array a, let $c(l, r)$ indicate how many distinct numbers are among $a_l, a_{l+1}, \ldots, a_r$. In particular, if $l > r$, define $c(l, r) = 0$.

You are given a string s of length n consisting of letters L and R only. Let a non-negative array a be called *good*, if the following conditions hold for $1 \le i \le n$:

- if $s_i = \mathrm{L}$, then $c(1, i - 1) = a_i$;
- if $s_i = \mathrm{R}$, then $c(i + 1, n) = a_i$.

If there is a good array a, print any of the good arrays. Otherwise, report that no such arrays exists.

Input

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

The second line of each test case contains a string s with a length n, containing only English uppercase letters L and R.

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

Output

For each test case, if a good array exists, print n non-negative integers: a good array a. Otherwise, print a single integer -1.

If there are multiple arrays a satisfying the conditions, you can output any of them.

input
4 3 LLR 3 RRL 4 RRLR 5 LLRLR

output

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

In the first test case, the array $[0, 1, 0]$ satisfies the conditions because:

- When $i = 1$, $s_i = \texttt{L}$, and $c(1, 0) = 0$;
- When $i = 2$, $s_i = \texttt{L}$, and $c(1, 1) = 1$, since there is only one distinct number in a_1;
- When $i = 3$, $s_i = \texttt{R}$, and $c(4, 3) = 0$.

In the second test case, another suitable answer is $[1, 1, 1]$.

In the third test case, it can be proven that there's no array satisfying the conditions.

I2. Kevin and Puzzle (Hard Version)

6 seconds, 512 megabytes

This is the hard version of the problem. The difference between the versions is that in this version, you need to count the number of good arrays. You can hack only if you solved all versions of this problem.

Kevin is visiting the Red Church, and he found a puzzle on the wall.

For an array a, let $c(l, r)$ indicate how many distinct numbers are among $a_l, a_{l+1}, \ldots, a_r$. In particular, if $l > r$, define $c(l, r) = 0$.

You are given a string s of length n consisting of letters $\texttt{L}$ and $\texttt{R}$ only. Let a non-negative array a be called *good*, if the following conditions hold for $1 \le i \le n$:

- if $s_i = \texttt{L}$, then $c(1, i - 1) = a_i$;
- if $s_i = \texttt{R}$, then $c(i + 1, n) = a_i$.

You need to count the number of good arrays a. Since the answer may be large, you only need to output the answer modulo $998\,244\,353$.

Input

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

The second line of each test case contains a string s with a length n, containing only English uppercase letters $\texttt{L}$ and $\texttt{R}$.

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

input

```
4
3
LLR
3
RRL
4
RRLR
5
LLRLR
```

output

```
1
2
0
1
```

All arrays satisfying the conditions can be found in the easy version of this problem.