A. Shape Perimeter

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

There is an m by m square stamp on an infinite piece of paper. Initially, the bottom-left corner of the square stamp is aligned with the bottom-left corner of the paper. You are given two integer sequences x and y, each of length n. For each step i from 1 to n, the following happens:

- Move the stamp x_i units to the right and y_i units upwards.
- Press the stamp onto the paper, leaving an m by m colored square at its current position.

Note that the elements of sequences x and y have a special constraint: $1 \le x_i, y_i \le m - 1$.

Note that you **do not** press the stamp at the bottom-left corner of the paper. Refer to the notes section for better understanding.

It can be proven that after all the operations, the colored shape on the paper formed by the stamp is a single connected region. Find the perimeter of this colored shape.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($1 \le n \le 100$, $2 \le m \le 100$) — the number of operations performed and the side length of the square stamp.

The i-th of the next n lines contains two integers x_i and y_i ($1 \le x_i, y_i \le m - 1$) — the distance that the stamp will be moved right and up during the i-th operation, respectively.

Note that there are **no** constraints on the sum of n over all test cases.

Output

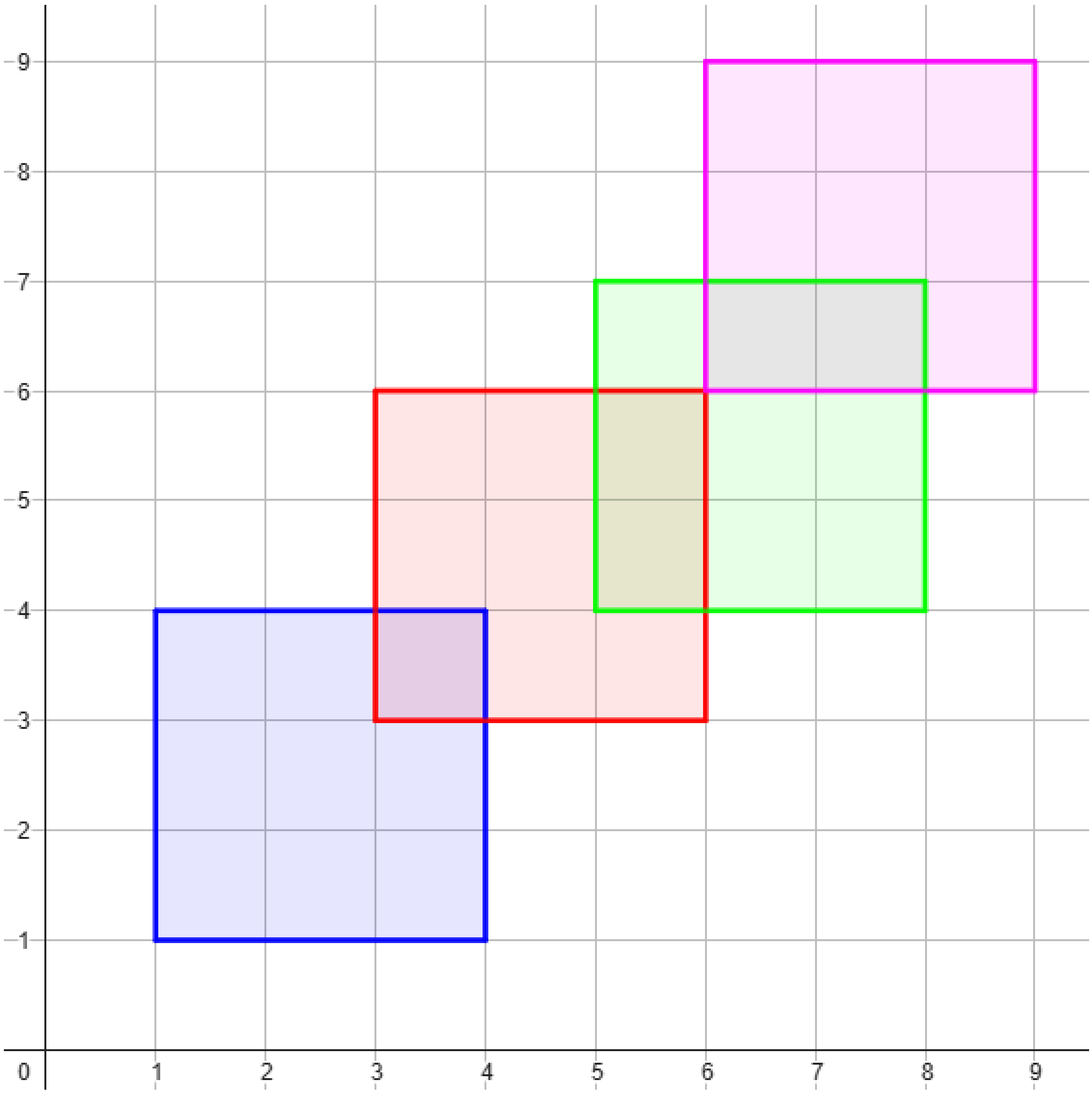
For each test case, output a single integer representing the perimeter of the colored shape on the paper.

Standard Input	Standard Output

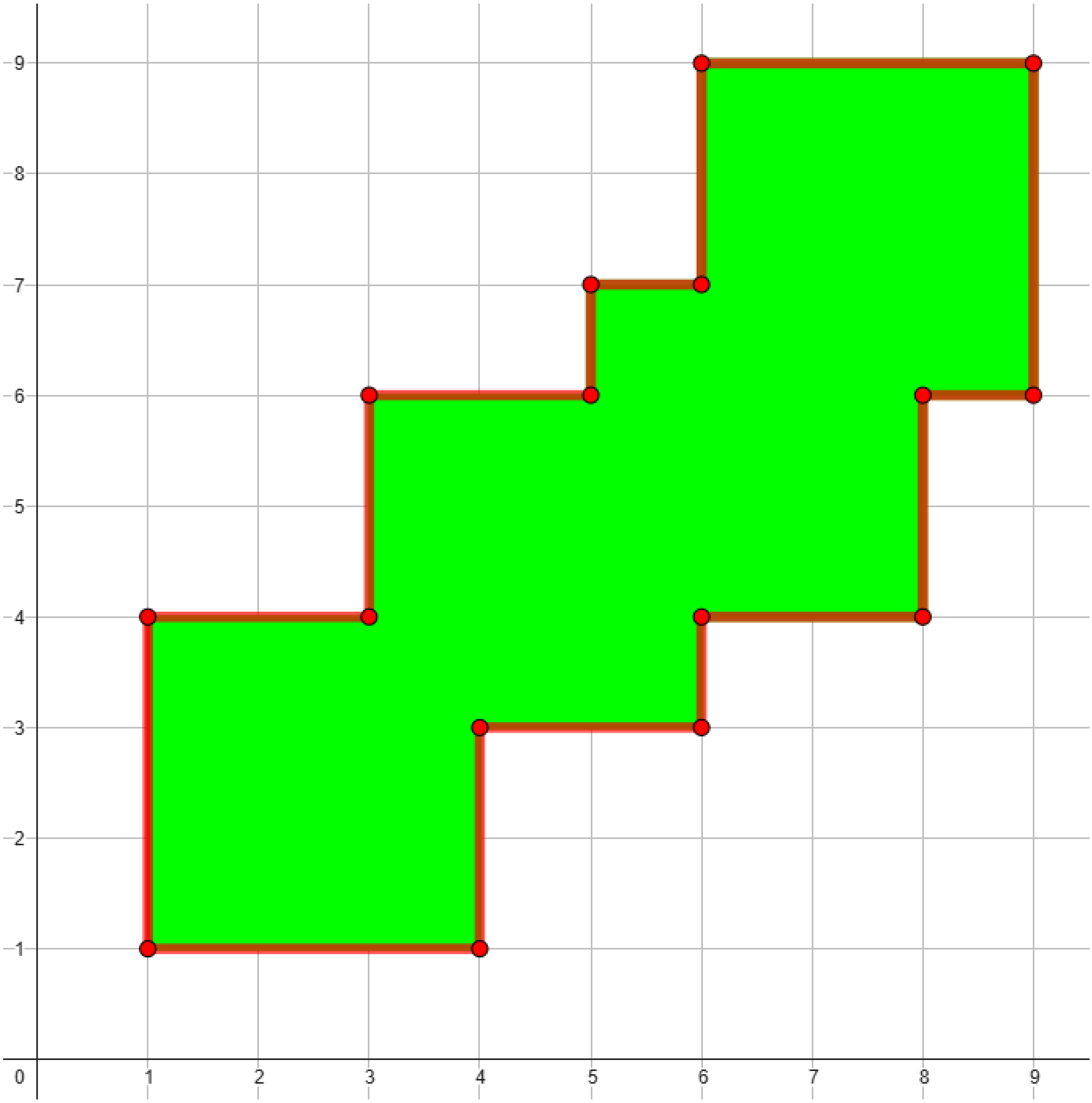
3 4 3 1 1 2 2 2 1 1 2 1 2 1 1 6 7 3 6 1 1 3 1 6 6 5 4 6 1	32 8 96

Note

In the first example, the stamp has a side length of 3 and is pressed 4 times at coordinates $(1, 1)$, $(3, 3)$, $(5, 4)$, and $(6, 6)$. The piece of paper looks like that afterwards:



Here, the square formed by the first press is colored blue, the second red, the third green, and the fourth purple. The combined shape, whose perimeter we need to calculate, looks like that:



From the diagram, it can be seen that this shape has a perimeter of 32.

B. Find the Permutation

```
Input file:    standard input
Output file:   standard output
Time limit:    1.5 seconds
Memory limit:  256 megabytes
```

You are given an undirected graph with n vertices, labeled from 1 to n. This graph encodes a hidden permutation* p of size n. The graph is constructed as follows:

- For every pair of integers $1 \le i < j \le n$, an undirected edge is added between vertex p_i and vertex p_j if and only if $p_i < p_j$. Note that the edge **is not added** between vertices i and j, but between the vertices of their respective elements. Refer to the notes section for better understanding.

Your task is to reconstruct and output the permutation p. It can be proven that permutation p can be uniquely determined.

*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).

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 500$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 1000$).

The i-th of the next n lines contains a string of n characters $g_{i,1}g_{i,2}\ldots g_{i,n}$ ($g_{i,j} = 0$ or $g_{i,j} = 1$) — the adjacency matrix. $g_{i,j} = 1$ if and only if there is an edge between vertex i and vertex j.

It is guaranteed that there exists a permutation p which generates the given graph. It is also guaranteed that the graph is undirected and has no self-loops, meaning $g_{i,j} = g_{j,i}$ and $g_{i,i} = 0$.

It is guaranteed that the sum of n over all test cases does not exceed 1000.

Output

For each test case, output n integers $p_1, p_2, \ldots, p_n$ representing the reconstructed permutation.

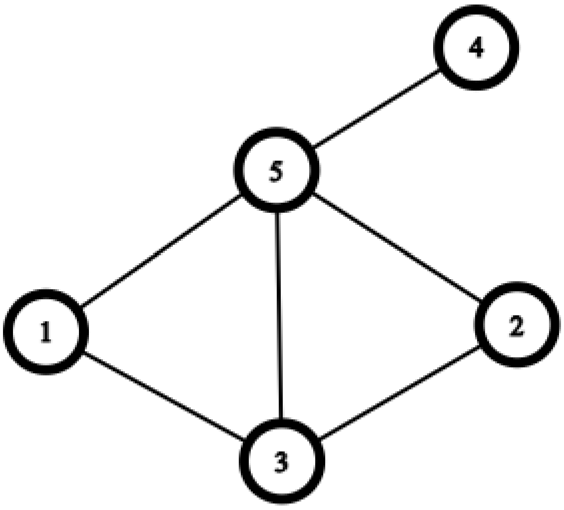
Standard Input	Standard Output
3 1 0 5 00101 00101 11001 00001 11110 6 000000 000000 000000	1 4 2 1 3 5 6 5 4 3 2 1

000000 000000 000000	

Note

In the first case $p = [1]$. Since there are no pairs $1 \le i < j \le n$, there are no edges in the graph.

The graph in the second case is shown below. For example, when we choose $i = 3$ and $j = 4$, we add an edge between vertices $p_i = 1$ and $p_j = 3$, because $p_i < p_j$. However, when we choose $i = 2$ and $j = 3$, $p_i = 2$ and $p_j = 1$, so $p_i < p_j$ doesn't hold. Therefore, we don't add an edge between 2 and 1.



In the third case, there are no edges in the graph, so there are no pairs of integers $1 \le i < j \le n$ such that $p_i < p_j$. Therefore, $p = [6, 5, 4, 3, 2, 1]$.

C. Palindromic Subsequences

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   512 megabytes
```

For an integer sequence $a = [a_1, a_2, \ldots, a_n]$, we define $f(a)$ as the length of the longest subsequence* of a that is a palindrome†.

Let $g(a)$ represent the number of subsequences of length $f(a)$ that are palindromes. In other words, $g(a)$ counts the number of palindromic subsequences in a that have the maximum length.

Given an integer n, your task is to find any sequence a of n integers that satisfies the following conditions:

- $1 \le a_i \le n$ for all $1 \le i \le n$.
- $g(a) > n$

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

---

* A sequence x is a subsequence of a sequence y if x can be obtained from y by the deletion of several (possibly, zero or all) element from arbitrary positions.

† A palindrome is a sequence that reads the same from left to right as from right to left. For example, $[1, 2, 1, 3, 1, 2, 1]$, $[5, 5, 5, 5]$, and $[4, 3, 3, 4]$ are palindromes, while $[1, 2]$ and $[2, 3, 3, 3, 3]$ are not.

Input

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

The first line of each test case contains a single integer n ($6 \le n \le 100$) — the length of the sequence.

Note that there are **no** constraints on the sum of n over all test cases.

Output

For each test case, output n integers $a_1, a_2, \ldots, a_n$, representing an array that satisfies the conditions.

If there are multiple solutions, you may output any of them.

Standard Input	Standard Output
3 6 9 15	1 1 2 3 1 2 7 3 3 7 5 3 7 7 3 15 8 8 8 15 5 8 1 15 5 8 15 15 15 8

Note

In the first example, one possible solution is $a = [1, 1, 2, 3, 1, 2]$. In this case, $f(a) = 3$ as the longest palindromic subsequence has length 3. There are 7 ways to choose a subsequence of length 3 that is a palindrome, as shown below:

1. $[a_1, a_2, a_5] = [1, 1, 1]$
2. $[a_1, a_3, a_5] = [1, 2, 1]$
3. $[a_1, a_4, a_5] = [1, 3, 1]$

4. $[a_2, a_3, a_5] = [1, 2, 1]$
5. $[a_2, a_4, a_5] = [1, 3, 1]$
6. $[a_3, a_4, a_6] = [2, 3, 2]$
7. $[a_3, a_5, a_6] = [2, 1, 2]$

Therefore, $g(a) = 7$, which is greater than $n = 6$. Hence, $a = [1, 1, 2, 3, 1, 2]$ is a valid solution.

In the second example, one possible solution is $a = [7, 3, 3, 7, 5, 3, 7, 7, 3]$. In this case, $f(a) = 5$. There are 24 ways to choose a subsequence of length 5 that is a palindrome. Some examples are $[a_2, a_4, a_5, a_8, a_9] = [3, 7, 5, 7, 3]$ and $[a_1, a_4, a_6, a_7, a_8] = [7, 7, 3, 7, 7]$. Therefore, $g(a) = 24$, which is greater than $n = 9$. Hence, $a = [7, 3, 3, 7, 5, 3, 7, 7, 3]$ is a valid solution.

In the third example, $f(a) = 7$ and $g(a) = 190$, which is greater than $n = 15$.

,

D. Unique Median

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 512 megabytes

An array b of m integers is called *good* if, **when it is sorted**, $b_{\left\lfloor \frac{m+1}{2} \right\rfloor} = b_{\left\lceil \frac{m+1}{2} \right\rceil}$. In other words, b is good if both of its medians are equal. In particular, $\left\lfloor \frac{m+1}{2} \right\rfloor = \left\lceil \frac{m+1}{2} \right\rceil$ when m is odd, so b is guaranteed to be good if it has an odd length.

You are given an array a of n integers. Calculate the number of good subarrays* in a.

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

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 10^5$) — the length of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le \mathbf{10}$) — the given array.

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

Output

For each test case, output a single integer representing the number of good subarrays in a.

Standard Input	Standard Output
3 4 1 1 1 1 5 1 10 2 3 3 10 6 3 2 3 5 3 4 2 3 5	10 11 42

Note

In the first case, every subarray is good since all its elements are equal to 1.

In the second case, an example of a good subarray is $b = [10, 2, 3, 3]$. When it is sorted, $b = [2, 3, 3, 10]$, so $b_{\left\lfloor \frac{4+1}{2} \right\rfloor} = b_{\left\lceil \frac{4+1}{2} \right\rceil} = b_2 = b_3 = 3$. Another example would be $b = [1, 10, 2]$. On the other hand, $b = [1, 10]$ is not good as its two medians are 1 and 10, which are not equal.

E. Nested Segments

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

A set A consisting of pairwise distinct segments $[l, r]$ with integer endpoints is called *good* if $1 \le l \le r \le n$, and for any pair of distinct segments $[l_i, r_i], [l_j, r_j]$ in A, exactly one of the following conditions holds:

- $r_i < l_j$ or $r_j < l_i$ (the segments do not intersect)
- $l_i \le l_j \le r_j \le r_i$ or $l_j \le l_i \le r_i \le r_j$ (one segment is fully contained within the other)

You are given a good set S consisting of m pairwise distinct segments $[l_i, r_i]$ with integer endpoints. You want to add as many additional segments to the set S as possible while ensuring that set S remains good.

Since this task is too easy, you need to determine the number of different ways to add the maximum number of additional segments to S, ensuring that the set remains good. Two ways are considered different if there exists a segment that is being added in one of the ways, but not in the other.

Formally, you need to find the number of good sets T of distinct segments, such that S is a subset of T and T has the maximum possible size. Since the result might be very large, compute the answer modulo $998\,244\,353$.

Input

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

The first line of each test case contains two integers n and m ($1 \le n \le 2 \cdot 10^5$, $0 \le m \le 2 \cdot 10^5$) — the maximum right endpoint of the segments, and the size of S.

The i-th of the next m lines contains two integers l_i and r_i ($1 \le l_i \le r_i \le n$) — the boundaries of the segments in set S.

It is guaranteed that the given set S is good, and the segments in set S are pairwise distinct.

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

Output

For each test case, output a single integer representing the number of different ways, modulo $998\,244\,353$, that you can add the maximum number of additional segments to set S while ensuring that set S remains good.

Standard Input	Standard Output
6 1 0 2 3 1 1 2 2 1 2 5 2 1 3 2 3	1 1 2 5 4 187997613

4 1 1 1 6 2 1 3 4 6 2300 0	

Note

In the first example, the only possible segment is $[1, 1]$, so $T = \{[1, 1]\}$ has the maximum size, and it is the only solution.

In the second example, it is not possible to add any additional segments to set S. Hence, the only way to add segments to S is adding nothing.

In the third example, it is possible to add 7 additional segments to S while ensuring that the set remains good. It can be proven that adding more than 7 additional segments to S is not possible. There are exactly 2 different ways to add these 7 segments to S, and their respective sets T are shown below:

- $\{[1, 1], [1, 3], [1, 4], [1, 5], [2, 2], [2, 3], [3, 3], [4, 4], [5, 5]\}$
- $\{[1, 1], [1, 3], [1, 5], [2, 2], [2, 3], [3, 3], [4, 4], [4, 5], [5, 5]\}$.

In the fourth example, there are exactly 5 different ways to add a maximum of 6 additional segments to S, and their respective sets T are shown below:

- $\{[1, 1], [1, 2], [1, 3], [1, 4], [2, 2], [3, 3], [4, 4]\}$
- $\{[1, 1], [1, 2], [1, 4], [2, 2], [3, 3], [3, 4], [4, 4]\}$
- $\{[1, 1], [1, 3], [1, 4], [2, 2], [2, 3], [3, 3], [4, 4]\}$
- $\{[1, 1], [1, 4], [2, 2], [2, 3], [2, 4], [3, 3], [4, 4]\}$
- $\{[1, 1], [1, 4], [2, 2], [2, 4], [3, 3], [3, 4], [4, 4]\}$

F1. Xor of Median (Easy Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   256 megabytes
```

This is the easy version of the problem. The difference between the versions is that in this version, the constraints on t, k, and m are lower. You can hack only if you solved all versions of this problem.

A sequence a of n integers is called *good* if the following condition holds:

- Let cnt_x be the number of occurrences of x in sequence a. For all pairs $0 \le i < j < m$, at least one of the following has to be true: $\text{cnt}_i = 0$, $\text{cnt}_j = 0$, or $\text{cnt}_i \le \text{cnt}_j$. In other words, if both i and j are present in sequence a, then the number of occurrences of i in a is less than or equal to the number of occurrences of j in a.

You are given integers n and m. Calculate the value of the bitwise XOR of the median* of all good sequences a of length n with $0 \le a_i < m$.

Note that the value of n can be very large, so you are given its binary representation instead.

---

* The median of a sequence a of length n is defined as the $\left\lfloor \frac{n+1}{2} \right\rfloor$-th smallest value in the sequence.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 50$). The description of the test cases follows.

The first line of each test case contains two integers k and m ($1 \le k \le 200$, $1 \le m \le 500$) — the number of bits in n and the upper bound on the elements in sequence a.

The second line of each test case contains a binary string of length k — the binary representation of n with no leading zeros.

It is guaranteed that the sum of k over all test cases does not exceed 200.

Output

For each test case, output a single integer representing the bitwise XOR of the median of all good sequences a of length n where $0 \le a_i < m$.

Standard Input	Standard Output
6 2 3 10 2 3 11 5 1 11101 7 9 1101011 17 34 11001010001010010	3 2 0 8 32 0

1 500 1	

Note

In the first example, $n = 10_2 = 2$ and $m = 3$. All possible sequences with elements less than m are: $[0, 0]$, $[0, 1]$, $[0, 2]$, $[1, 0]$, $[1, 1]$, $[1, 2]$, $[2, 0]$, $[2, 1]$, $[2, 2]$. All of them are good, so the answer is: $0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 2 = 3$.

In the second example, $n = 11_2 = 3$ and $m = 3$. Some good sequences are $[2, 2, 2]$, $[1, 0, 1]$, and $[2, 0, 1]$. However, a sequence $[2, 0, 0]$ is not good, because $\text{cnt}_0 = 2$, $\text{cnt}_2 = 1$. Therefore, if we set $i = 0$ and $j = 2$, $i < j$ holds, but $\text{cnt}_i \le \text{cnt}_j$ does not.

,

F2. Xor of Median (Hard Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   256 megabytes
```

This is the hard version of the problem. The difference between the versions is that in this version, the constraints on t, k, and m are higher. You can hack only if you solved all versions of this problem.

A sequence a of n integers is called *good* if the following condition holds:

- Let cnt_x be the number of occurrences of x in sequence a. For all pairs $0 \le i < j < m$, at least one of the following has to be true: $\text{cnt}_i = 0$, $\text{cnt}_j = 0$, or $\text{cnt}_i \le \text{cnt}_j$. In other words, if both i and j are present in sequence a, then the number of occurrences of i in a is less than or equal to the number of occurrences of j in a.

You are given integers n and m. Calculate the value of the bitwise XOR of the median* of all good sequences a of length n with $0 \le a_i < m$.

Note that the value of n can be very large, so you are given its binary representation instead.

---

* The median of a sequence a of length n is defined as the $\left\lfloor \frac{n+1}{2} \right\rfloor$-th smallest value in the sequence.

Input

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

The first line of each test case contains two integers k and m ($1 \le k \le 2 \cdot 10^5$, $1 \le m \le 10^9$) — the number of bits in n and the upper bound on the elements in sequence a.

The second line of each test case contains a binary string of length k — the binary representation of n with no leading zeros.

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

Output

For each test case, output a single integer representing the bitwise XOR of the median of all good sequences a of length n where $0 \le a_i < m$.

Standard Input	Standard Output
6 2 3 10 2 3 11 5 1 11101 7 9 1101011 17 34 11001010001010010	3 2 0 8 32 0

1 1000000000 1	

Note

In the first example, $n = 10_2 = 2$ and $m = 3$. All possible sequences with elements less than m are: $[0, 0]$, $[0, 1]$, $[0, 2]$, $[1, 0]$, $[1, 1]$, $[1, 2]$, $[2, 0]$, $[2, 1]$, $[2, 2]$. All of them are good, so the answer is: $0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 1 \oplus 2 = 3$.

In the second example, $n = 11_2 = 3$ and $m = 3$. Some good sequences are $[2, 2, 2]$, $[1, 0, 1]$, and $[2, 0, 1]$. However, a sequence $[2, 0, 0]$ is not good, because $\mathrm{cnt}_0 = 2$, $\mathrm{cnt}_2 = 1$. Therefore, if we set $i = 0$ and $j = 2$, $i < j$ holds, but $\mathrm{cnt}_i \le \mathrm{cnt}_j$ does not.