Codeforces Round 1032 (Div. 3)

A. Letter Home

1 second, 256 megabytes

You are given an array of distinct integers $x_1, x_2, \ldots, x_n$ and an integer s.

Initially, you are at position $pos = s$ on the X axis. In one step, you can perform exactly one of the following two actions:

- Move from position pos to position $pos + 1$.
- Move from position pos to position $pos - 1$.

A sequence of steps will be considered successful if, during the entire journey, you visit each position x_i on the X axis at least once. Note that the initial position $pos = s$ is also considered visited.

Your task is to determine the minimum number of steps in any successful sequence of steps.

Input

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

The first line of each test case contains two integers n and s ($1 \le n \le 10, 1 \le s \le 100$) — the number of positions to visit and the starting position.

The second line of each test case contains n integers $x_1, x_2, \ldots, x_n$ ($1 \le x_i \le 100$). It is guaranteed that for all $1 \le i < n$, it holds that $x_i < x_{i+1}$.

Output

For each test case, output the minimum number of steps in any successful sequence of steps.

input

```
12
1 1
1
1 2
1
1 1
2
2 1
2 3
2 2
1 3
2 3
1 2
3 1
1 2 3
3 2
1 3 4
3 3
1 2 3
4 3
1 2 3 10
5 5
1 2 3 6 7
6 6
1 2 3 9 10 11
```

output

```
0
1
1
2
3
2
2
4
2
11
8
15
```

In the first test case, no steps need to be taken, so the only visited position will be 1.

In the second test case, the following path can be taken: $2 \to 1$. The number of steps is 1.

In the third test case, the following path can be taken: $1 \to 2$. The number of steps is 1.

In the fifth test case, the following path can be taken: $2 \to 1 \to 2 \to 3$. The number of steps is 3.

B. Above the Clouds

2 seconds, 256 megabytes

You are given a string s of length n, consisting of lowercase letters of the Latin alphabet. Determine whether there exist three **non-empty** strings a, b, and c such that:

- $a + b + c = s$, meaning the concatenation* of strings a, b, and c equals s.
- The string b is a substring† of the string $a + c$, which is the concatenation of strings a and c.

*Concatenation of strings a and b is defined as the string $a + b = a_1a_2 \ldots a_pb_1b_2 \ldots b_q$, where p and q are the lengths of strings a and b, respectively. For example, the concatenation of the strings "code" and "forces" is "codeforces".

†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 consists of 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 ($3 \le n \le 10^5$) — the length of the string s.

The second line of each test case contains the string s of length n, consisting of lowercase letters of the Latin alphabet.

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

Output

For each test case, output "Yes" if there exist three non-empty strings a, b, and c that satisfy the conditions, and "No" otherwise.

You may output the answer in any case (upper or lower). For example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as positive answers.

input

```
12
3
aaa
3
aba
3
aab
4
abca
4
abba
4
aabb
5
abaca
5
abcda
5
abcba
6
abcbbf
6
abcdaa
3
abb
```

output

```
Yes
No
Yes
No
Yes
Yes
Yes
No
Yes
Yes
Yes
Yes
```

In the first test case, there exist unique non-empty strings a, b, and c such that $a + b + c = s$. These are the strings $a =$ "a", $b =$ "a", and $c =$ "a". The concatenation of strings a and c equals $a + c =$ "aa". The string b is a substring of this string.

In the sixth test case, one can choose $a =$ "a", $b =$ "ab", and $c =$ "b". The concatenation of strings a and c equals $a + c =$ "ab". The string b is a substring of this string.

In the seventh test case, one can choose $a =$ "ab", $b =$ "a", and $c =$ "ca". The concatenation of strings a and c equals $a + c =$ "abca". The string b is a substring of this string.

C. Those Who Are With Us

1 second, 256 megabytes

You are given a matrix of integers with n rows and m columns. The cell at the intersection of the i-th row and the j-th column contains the number a_{ij}.

You can perform the following operation **exactly once**:

- Choose two numbers $1 \le r \le n$ and $1 \le c \le m$.
- For all cells (i, j) in the matrix such that $i = r$ or $j = c$, decrease a_{ij} by one.

You need to find the minimal possible maximum value in the matrix a after performing exactly one such operation.

Input

Each test consists of 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 two integers n and m ($1 \le n \cdot m \le 10^5$) — the number of rows and columns in the matrix.

The next n lines of each test case describe the matrix a. The i-th line contains m integers $a_{i1}, a_{i2}, \ldots, a_{im}$ $(1 \le a_{ij} \le 100)$ — the elements in the i-th row of the matrix.

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

Output

For each test case, output the minimum maximum value in the matrix a after performing exactly one operation.

input

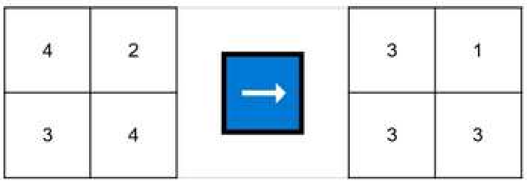
```
10
1 1
1
1 2
1 2
2 1
2
1
2 2
4 2
3 4
3 4
1 2 3 2
3 2 1 3
2 1 3 2
4 3
1 5 1
3 1 3
5 5 5
3 5 1
4 4
1 3 3 2
2 3 2 2
1 2 2 1
3 3 2 3
2 2
2 2
1 2
3 2
1 2
2 1
1 2
3 3
2 1 1
1 2 1
1 1 2
```

output

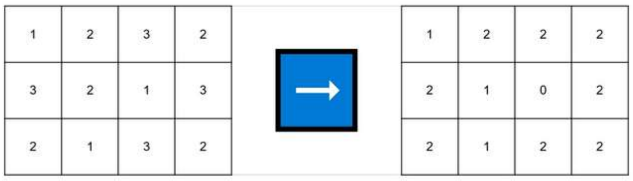
```
0
1
1
3
2
4
3
1
1
2
```

In the first three test cases, you can choose $r = 1$ and $c = 1$.

In the fourth test case, you can choose $r = 1$ and $c = 2$.



In the fifth test case, you can choose $r = 2$ and $c = 3$.



In the sixth test case, you can choose $r = 3$ and $c = 2$.

D. 1709

2 seconds, 256 megabytes

You are given two arrays of integers $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$. It is guaranteed that each integer from 1 to $2 \cdot n$ appears in exactly one of the arrays.

You need to perform a certain number of operations (possibly zero) so that **both** of the following conditions are satisfied:

- For each $1 \leq i < n$, it holds that $a_i < a_{i+1}$ and $b_i < b_{i+1}$.
- For each $1 \leq i \leq n$, it holds that $a_i < b_i$.

During each operation, you can perform exactly one of the following three actions:

1. Choose an index $1 \leq i < n$ and swap the values a_i and a_{i+1}.
2. Choose an index $1 \leq i < n$ and swap the values b_i and b_{i+1}.
3. Choose an index $1 \leq i \leq n$ and swap the values a_i and b_i.

You do not need to minimize the number of operations, but the total number must not exceed 1709. Find any sequence of operations that satisfies **both** conditions.

Input

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

The first line of each test case contains a single integer n ($1 \leq n \leq 40$) — the length of the arrays a and b.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 2 \cdot n$).

The third line of each test case contains n integers $b_1, b_2, \ldots, b_n$ ($1 \leq b_i \leq 2 \cdot n$).

It is guaranteed that each integer from 1 to $2 \cdot n$ appears either in array a or in array b.

Output

For each test case, output the sequence of operations.

In the first line for each test case, output the number of operations k. Note that $0 \leq k \leq 1709$.

In the following k lines for each test case, output the operations themselves:

- If you want to swap the values a_i and a_{i+1}, output two integers 1 and i. Note that $1 \leq i < n$.
- If you want to swap the values b_i and b_{i+1}, output two integers 2 and i. Note that $1 \leq i < n$.
- If you want to swap the values a_i and b_i, output two integers 3 and i. Note that $1 \leq i \leq n$.

It can be shown that under the given constraints, a solution always exists.

input
```
6
1
1
2
1
2
1
2
1 3
4 2
2
1 4
3 2
3
6 5 4
3 2 1
3
5 3 4
2 6 1
```

output
```
0
1
3 1
1
2 1
1
3 2
9
3 1
3 2
3 3
1 1
2 1
2 2
1 2
1 1
2 1
6
2 2
1 1
1 2
2 1
3 1
3 2
```

In the first test case, $a_1 < b_1$, so no operations need to be applied.

In the second test case, $a_1 > b_1$. After applying the operation, these values will be swapped.

In the third test case, after applying the operation, $a = [1, 3]$ and $b = [2, 4]$.

In the fourth test case, after applying the operation, $a = [1, 2]$ and $b = [3, 4]$.

E. Sponsor of Your Problems

2 seconds, 256 megabytes

For two integers a and b, we define $f(a, b)$ as the number of positions in the decimal representation of the numbers a and b where their digits are the same. For example, $f(12, 21) = 0$, $f(31, 37) = 1$, $f(19891, 18981) = 2$, $f(54321, 24361) = 3$.

You are given two integers l and r of the **same** length in decimal representation. Consider all integers $l \leq x \leq r$. Your task is to find the minimum value of $f(l, x) + f(x, r)$.

Input

Each test 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 the test cases follows.

Each test case consists of a single line containing two integers l and r ($1 \leq l \leq r < 10^9$).

It is guaranteed that the numbers l and r have the same length in decimal representation and do not have leading zeros.

Output

For each test case, output the minimum value of $f(l, x) + f(x, r)$ among all integer values $l \le x \le r$.

input

```
14
1 1
2 3
4 6
15 16
17 19
199 201
899 999
1990 2001
6309 6409
12345 12501
19987 20093
746814 747932
900990999 900991010
999999999 999999999
```

output

```
2
1
0
3
2
2
1
3
3
4
3
5
12
18
```

In the first test case, you can choose $x = 1$. Then $f(1, 1) + f(1, 1) = 1 + 1 = 2$.

In the second test case, you can choose $x = 2$. Then $f(2, 2) + f(2, 3) = 1 + 0 = 1$.

In the third test case, you can choose $x = 5$. Then $f(4, 5) + f(5, 6) = 0 + 0 = 0$.

In the fourth test case, you can choose $x = 15$. Then $f(15, 15) + f(15, 16) = 2 + 1 = 3$.

In the fifth test case, you can choose $x = 18$. Then $f(17, 18) + f(18, 19) = 1 + 1 = 2$.

In the sixth test case, you can choose $x = 200$. Then $f(199, 200) + f(200, 201) = 0 + 2 = 2$.

In the seventh test case, you can choose $x = 900$. Then $f(899, 900) + f(900, 999) = 0 + 1 = 1$.

In the eighth test case, you can choose $x = 1992$. Then $f(1990, 1992) + f(1992, 2001) = 3 + 0 = 3$.

F. Yamakasi

3 seconds, 256 megabytes

You are given an array of integers $a_1, a_2, \ldots, a_n$ and two integers s and x. Count the number of subsegments of the array whose sum of elements equals s and whose maximum value equals x.

More formally, count the number of pairs $1 \le l \le r \le n$ such that:

- $a_l + a_{l+1} + \ldots + a_r = s$.
- $\max(a_l, a_{l+1}, \ldots, a_r) = x$.

Input

Each test consists of 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 three integers n, s, and x ($1 \le n \le 2 \cdot 10^5$, $-2 \cdot 10^{14} \le s \le 2 \cdot 10^{14}$, $-10^9 \le x \le 10^9$).

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$).

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

Output

For each test case, output the number of subsegments of the array whose sum of elements equals s and whose maximum value equals x.

input

```
9
1 0 0
0
1 -2 -1
-2
3 -1 -1
-1 1 -1
6 -3 -2
-1 -1 -1 -2 -1 -1
8 3 2
2 2 -1 -2 3 -1 2 2
9 6 3
1 2 3 1 2 3 1 2 3
13 7 3
0 -1 3 3 3 -2 1 2 2 3 -1 0 3
2 -2 -1
-2 -1
2 -2 -1
-1 -2
```

output

```
1
0
2
0
2
7
8
0
0
```

In the first test case, the suitable subsegment is $l = 1, r = 1$.

In the third test case, the suitable subsegments are $l = 1, r = 1$ and $l = 3, r = 3$.

In the fifth test case, the suitable subsegments are $l = 1, r = 3$ and $l = 6, r = 8$.

In the sixth test case, the suitable subsegments are those for which $r = l + 2$.

In the seventh test case, the following subsegments are suitable:

- $l = 1, r = 7$.
- $l = 2, r = 7$.
- $l = 3, r = 6$.
- $l = 4, r = 8$.
- $l = 7, r = 11$.
- $l = 7, r = 12$.
- $l = 8, r = 10$.
- $l = 9, r = 13$.

G. Gangsta

2 seconds, 256 megabytes

You are given a binary string $s_1 s_2 \ldots s_n$ of length n. A string s is called binary if it consists only of zeros and ones.

For a string p, we define the function $f(p)$ as the maximum number of occurrences of any character in the string p. For example, $f(00110) = 3$, $f(01) = 1$.

You need to find the sum $f(s_l s_{l+1} \ldots s_r)$ for all pairs $1 \le l \le r \le n$.

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the binary string.

The second line of each test case contains a string of length n, consisting of 0s and 1s — the binary string s.

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

Output

For each test case, output the sum $f(s_l s_{l+1} \dots s_r)$ for all pairs $1 \le l \le r \le n$.

input

```
6
1
0
2
01
4
0110
6
110001
8
10011100
11
01011011100
```

output

```
1
3
14
40
78
190
```

In the first test case, the string s has one substring, and the value $f(0) = 1$.

In the second test case, all substrings of the string s are 0, 01, 1. And the answer is $1 + 1 + 1 = 3$, respectively.

In the third test case, all substrings of the string s are 0, 01, 011, 0110, 1, 11, 110, 1, 10, 0. And the answer is $1 + 1 + 2 + 2 + 1 + 2 + 2 + 1 + 1 + 1 = 14$, respectively.

H. Ice Baby

2 seconds, 512 megabytes

The longest non-decreasing subsequence of an array of integers $a_1, a_2, \dots, a_n$ is the longest sequence of indices $1 \le i_1 < i_2 < \dots < i_k \le n$ such that $a_{i_1} \le a_{i_2} \le \dots \le a_{i_k}$. The length of the sequence is defined as the number of elements in the sequence. For example, the length of the longest non-decreasing subsequence of the array $a = [3, 1, 4, 1, 2]$ is 3.

You are given two arrays of integers $l_1, l_2, \dots, l_n$ and $r_1, r_2, \dots, r_n$. For each $1 \le k \le n$, solve the following problem:

- Consider all arrays of integers a of length k, such that for each $1 \le i \le k$, it holds that $l_i \le a_i \le r_i$. Find the maximum length of the longest non-decreasing subsequence among all such arrays.

Input

Each test consists of 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 ($1 \le n \le 2 \cdot 10^5$) — the length of the arrays l and r.

The next n lines of each test case contain two integers l_i and r_i ($1 \le l_i \le r_i \le 10^9$).

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

Output

For each test case, output n integers: for each k from 1 to n, output the maximum length of the longest non-decreasing subsequence among all suitable arrays.

input

```
6
1
1 1
2
3 4
1 2
4
4 5
3 4
1 3
3 3
8
6 8
4 6
3 5
5 5
3 4
1 3
2 4
3 3
5
1 2
6 8
4 5
2 3
3 3
11
35 120
66 229
41 266
98 164
55 153
125 174
139 237
30 72
138 212
109 123
174 196
```

output

```
1
1 1
1 2 2 3
1 2 2 3 3 3 4 5
1 2 2 2 3
1 2 3 4 5 6 7 7 8 8 9
```

In the first test case, the only possible array is $a = [1]$. The length of the longest non-decreasing subsequence of this array is 1.

In the second test case, for $k = 2$, no matter how we choose the values of a_1 and a_2, the condition $a_1 > a_2$ will always hold. Therefore, the answer for $k = 2$ will be 1.

In the third test case, for $k = 4$, we can choose the array $a = [5, 3, 3, 3]$. The length of the longest non-decreasing subsequence of this array is 3.

In the fourth test case, for $k = 8$, we can choose the array $a = [7, 5, 3, 5, 3, 3, 3, 3]$. The length of the longest non-decreasing subsequence of this array is 5.

In the fifth test case, for $k = 5$, we can choose the array $a = [2, 8, 5, 3, 3]$. The length of the longest non-decreasing subsequence of this array is 3.