A. Serval and String Theory

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

A string r consisting only of lowercase Latin letters is called *universal* if and only if r is lexicographically smaller * than the reversal† of r.

You are given a string s consisting of n lowercase Latin letters. You are required to make s *universal*. To achieve this, you can perform the following operation on s **at most** k times:

- Choose two indices i and j ($1 \le i, j \le n$), then swap s_i and s_j. Note that if $i = j$, you do nothing.

Determine whether you can make s *universal* by performing the above operation at most k times.

---

*A string a is lexicographically smaller than a string b of the same length, if and only if the following holds:

- in the first position where a and b differ, the string a has a letter that appears earlier in the alphabet than the corresponding letter in b.

†The reversal of a string r is the string obtained by writing r from right to left. For example, the reversal of the string `abcad` is `dacba`.

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 two integers n and k ($1 \le n \le 100, 0 \le k \le 10^4$) — the length of the string s, and the maximum number of operations you can perform.

The second line contains a string s consisting of n lowercase Latin letters.

Output

For each test case, print "YES" if it is possible to make s *universal* by performing the operation at most k times. Otherwise, print "NO".

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.

Standard Input	Standard Output
8 1 10000 a 3 3 rev 6 0 string 6 0 theory 9 2 universal 19 0	NO YES NO YES YES NO YES NO

codeforcesecrofedoc 19 1 codeforcesecrofedoc 3 1 zzz	

Note

In the first test case, s will keep the same after any operations. However, the reversal of `a` is still `a`, so it is impossible to make s *universal*.

In the second test case, the string `rev` is lexicographically smaller than `ver`. Thus, s is already *universal*.

In the fifth test case, you can perform the operations as follows:

1. Swap s_4 and s_7. After this operation, s becomes `uniserval`;
2. Swap s_1 and s_3. After this operation, s becomes `inuserval`.

And the string `inuserval` is *universal*.

,

B. Serval and Final MEX

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

You are given an array a consisting of $n \geq 4$ non-negative integers.

You need to perform the following operation on a until its length becomes 1:

- Select two indices l and r ($1 \leq l < r \leq |a|$), and replace the subarray $[a_l, a_{l+1}, \ldots, a_r]$ with a single integer $\operatorname{mex}([a_l, a_{l+1}, \ldots, a_r])$, where $\operatorname{mex}(b)$ denotes the minimum excluded (MEX)* of the integers in b. In other words, let $x = \operatorname{mex}([a_l, a_{l+1}, \ldots, a_r])$, the array a will become $[a_1, a_2, \ldots, a_{l-1}, x, a_{r+1}, a_{r+2}, \ldots, a_{|a|}]$. Note that the length of a decreases by $(r - l)$ after this operation.

Serval wants the final element in a to be 0. Help him!

More formally, you have to find a sequence of operations, such that after performing these operations in order, the length of a becomes 1, and the final element in a is 0.

It can be shown that at least one valid operation sequence exists under the constraints of the problem, and the length of any valid operation sequence does not exceed n.

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

*The minimum excluded (MEX) of a collection of integers $b_1, b_2, \ldots, b_k$ is defined as the smallest non-negative integer x which does not occur in the collection b.

Input

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

The first line of each test case contains a single integer n ($4 \leq n \leq 5000$) — the length of the array a.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq n$) — the elements of the array a.

It is guaranteed that the sum of n over all test cases does not exceed 5000.

Output

For each test case, output a single integer k ($0 \leq k \leq n$) in the first line of output — the length of the operation sequence.

Then, output k lines, the i-th line containing two integers l_i and r_i ($1 \leq l_i < r_i \leq |a|$) — the two indices you choose in the i-th operation, where $|a|$ denotes the length of the array before this operation.

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

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

0 1 0 0 1 6 0 0 0 0 0 0 6 5 4 3 2 1 0 4 0 0 1 1 4 1 0 0 0	1 2 1 2 1 2 4 5 6 3 4 1 2 1 3 3 4 5 4 5 1 4 2 1 2 1 3 2 2 4 1 2

Note

In the first test case, since $\operatorname{mex}([1,2,3,4]) = 0$, after the only operation, the array becomes $[0]$.

In the second test case, the array a changes as follows:

$$[\underline{0,1},0,0,1] \to [\underline{2,0},0,1] \to [\underline{1,0},1] \to [\underline{2,1}] \to [0].$$

In the third test case, the array a changes as follows:

$$[0,0,0,0,\underline{0,0}] \to [0,0,\underline{0,0},1] \to [\underline{0,0},1,1] \to [\underline{1,1,1}] \to [0].$$

,

C. Serval and The Formula

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

You are given two positive integers x and y ($1 \le x, y \le 10^9$).

Find a **non-negative** integer $k \le 10^{18}$, such that $(x + k) + (y + k) = (x + k) \oplus (y + k)$ holds*, or determine that such an integer does not exist.

*$\oplus$ denotes the bitwise XOR operation.

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 only line of each test case contains two integers x and y ($1 \le x, y \le 10^9$) — the given integers.

Output

For each test case, output a single integer k ($0 \le k \le 10^{18}$) — the integer you found. Print -1 if it is impossible to find such an integer.

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

Standard Input	Standard Output
5 2 5 6 6 19 10 1024 4096 1198372 599188	0 -1 1 1024 28

Note

In the first test case, since $(2 + 0) + (5 + 0) = (2 + 0) \oplus (5 + 0) = 7$, $k = 0$ is a possible answer. Note that $k = 4$ is also a possible answer because $(2 + 4) + (5 + 4) = (2 + 4) \oplus (5 + 4) = 15$.

In the second test case, $(x + k) \oplus (y + k) = (6 + k) \oplus (6 + k) = 0$. However, $(x + k) + (y + k) > 0$ holds for every $k \ge 0$, implying that such an integer k does not exist.

,

D. Serval and Kaitenzushi Buffet

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

Serval has just found a Kaitenzushi buffet restaurant. Kaitenzushi means that there is a conveyor belt in the restaurant, delivering plates of sushi in front of the customer, Serval.

In this restaurant, each plate contains exactly k pieces of sushi and the i-th plate has a *deliciousness* d_i. Serval will have a meal in this restaurant for n minutes, and within the n minutes, he must eat up **all** the pieces of sushi he took from the belt.

Denote the counter for uneaten taken pieces of sushi as r. Initially, $r = 0$. In the i-th minute ($1 \leq i \leq n$), only the i-th plate of sushi will be delivered in front of Serval, and he can do **one** of the following:

- Take the i-th plate of sushi (whose *deliciousness* is d_i) from the belt, and r will be increased by k;
- Eat one uneaten piece of sushi that he took from the belt before, and r will be decreased by 1. Note that you can do this only if $r > 0$;
- Or, do nothing, and r will remain unchanged.

Note that after the n minutes, the value of r **must** be 0.

Serval wants to maximize the sum of the *deliciousnesses* of all the plates he took. Help him find it out!

Input

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

The first line of each test case contains two integers n and k ($1 \leq k < n \leq 2 \cdot 10^5$) — the number of minutes of the meal and the number of sushi pieces in each plate.

The second line contains n integers $d_1, d_2, \ldots, d_n$ ($1 \leq d_i \leq 10^9$) — the *deliciousness* of each plate.

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

Output

For each test case, print a single integer — the maximum possible sum of the *deliciousnesses* of all the plates Serval took.

Standard Input	Standard Output
5 5 2 3 6 4 1 2 7 1 3 1 4 1 5 9 2 4 3 4 3 2 1 6 2 1 3 5 2 4 6 6 1	6 16 4 6 3000000000

1000000000 1 1000000000 1 1000000000 1	

Note

In the first test case, it can be shown that Serval can eat up at most one plate of sushi. Since the second plate of sushi has the greatest *deliciousness* 6 among all the plates, he will take it from the belt in the second minute, and then eat it up in the following 2 minutes.

Minute	1	2	3	4	5
Action	—	Take	Eat	Eat	—
r after action	0	2	1	0	0
Deliciousnesses gained	0	6	6	6	6

In the second test case, it can be shown that it is optimal for Serval to eat up the first, third, and sixth plates of sushi. The sum of the *deliciousnesses* of these plates is $3 + 4 + 9 = 16$.

Minute	1	2	3	4	5	6	7
Action	Take	Eat	Take	Eat	—	Take	Eat
r after action	1	0	1	0	0	1	0
Deliciousnesses gained	3	3	7	7	7	16	16

,

E. Serval and Modulo

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

There is an array a consisting of n non-negative integers and a *magic number* k ($k \geq 1$, k is an integer). Serval has constructed another array b of length n, where $b_i = a_i \bmod k$ holds* for all $1 \leq i \leq n$. Then, he **shuffled** b.

You are given the two arrays a and b. Find a possible *magic number* k. However, there is a small possibility that Serval fooled you, and such an integer does not exist. In this case, output -1.

It can be shown that, under the constraints of the problem, if such an integer k exists, then there exists a valid answer no more than 10^9. And you need to guarantee that $k \leq 10^9$ in your output.

*$a_i \bmod k$ denotes the remainder from dividing a_i by k.

Input

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

The first line of each test case contains an integer n ($1 \leq n \leq 10^4$) — the length of the array a.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \leq a_i \leq 10^6$) — the elements of the array a.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($0 \leq b_i \leq 10^6$) — the elements of the array b.

It is guaranteed that the sum of n over all test cases does not exceed 10^4.

Output

For each test case, output a single integer k ($1 \leq k \leq 10^9$) — the *magic number* you found. Print -1 if it is impossible to find such an integer.

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

Standard Input	Standard Output
5 4 3 5 2 7 0 1 1 1 5 3 1 5 2 4 1 2 3 4 5 6 2 3 4 7 8 9 1 2 3 6 7 8 5 21 22 25 28 20 0 1 2 1 0 6	2 31415926 -1 4 -1

1 1 2 3 5 8 0 0 1 1 0 0	

Note

In the first test case, if $k \geq 3$, then $2 = a_3 \bmod k$ should be in the array b, which leads to a contradiction. For $k = 1$, $[a_1 \bmod k, a_2 \bmod k, a_3 \bmod k, a_4 \bmod k] = [0, 0, 0, 0]$, which cannot be shuffled to b. For $k = 2$, $[a_1 \bmod k, a_2 \bmod k, a_3 \bmod k, a_4 \bmod k] = [1, 1, 0, 1]$, which can be shuffled to b. Thus, the only possible answer is $k = 2$.

In the second test case, note that b can be obtained by shuffling a. Thus, all integers between 6 and 10^9 are possible answers.

In the third test case, it can be shown that such k does not exist. Got fooled by Serval!

,

F1. Serval and Colorful Array (Easy Version)

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

This is the easy version of the problem. The difference between the versions is that in this version, $n \leq 3000$. You can hack only if you solved all versions of this problem.

Serval has a magic number k ($k \geq 2$). We call an array r *colorful* if and only if:

- The length of r is k, and
- Each integer between 1 and k appears **exactly once** in r.

You are given an array a consisting of n integers between 1 and k. It is guaranteed that each integer between 1 and k appears in a at least once. You can perform the following operation on a:

- Choose an index i ($1 \leq i < n$), then swap a_i and a_{i+1}.

Find the minimum number of operations needed to make at least one subarray* of a *colorful*. It can be shown that this is always possible under the constraints of the problem.

---

* An array b is a subarray of an array a if b can be obtained from a 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 ($1 \leq t \leq 1000$). The description of the test cases follows.

The first line of each test case contains two integers n and k ($2 \leq k \leq n \leq 3000$) — the length of the array a and Serval's magic number.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq k$) — the elements of the array a. It is guaranteed that each integer between 1 and k appears in a at least once.

It is guaranteed that the sum of n over all test cases does not exceed 3000.

Output

For each test case, output a single integer — the minimum number of operations needed to make at least one subarray of a *colorful*.

Standard Input	Standard Output
6 3 2 1 2 1 7 3 2 1 1 3 1 1 2 6 3 1 1 2 2 2 3 6 3 1 2 2 2 2 3 10 5	0 1 2 3 4 5

5 1 3 1 1 2 2 4 1 3 9 4 1 2 3 3 3 3 3 2 4	

Note

In the first test case, since the subarrays $[a_1, a_2] = [1, 2]$ and $[a_2, a_3] = [2, 1]$ are already *colorful*, we do not need to perform any operations. Thus, the answer is 0.

In the second test case, we can swap a_1 and a_2 to obtain $[1, \underline{2, 1, 3}, 1, 1, 2]$, which has a *colorful* subarray $[a_2, a_3, a_4] = [2, 1, 3]$. And the given array initially does not have any *colorful* subarrays, so the answer is 1.

,

F2. Serval and Colorful Array (Hard Version)

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

This is the hard version of the problem. The difference between the versions is that in this version, $n \leq 4 \cdot 10^5$. You can hack only if you solved all versions of this problem.

Serval has a magic number k ($k \geq 2$). We call an array r *colorful* if and only if:

- The length of r is k, and
- Each integer between 1 and k appears **exactly once** in r.

You are given an array a consisting of n integers between 1 and k. It is guaranteed that each integer between 1 and k appears in a at least once. You can perform the following operation on a:

- Choose an index i ($1 \leq i < n$), then swap a_i and a_{i+1}.

Find the minimum number of operations needed to make at least one subarray* of a *colorful*. It can be shown that this is always possible under the constraints of the problem.

---

* An array b is a subarray of an array a if b can be obtained from a 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 ($1 \leq t \leq 1000$). The description of the test cases follows.

The first line of each test case contains two integers n and k ($2 \leq k \leq n \leq 4 \cdot 10^5$) — the length of the array a and Serval's magic number.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq k$) — the elements of the array a. It is guaranteed that each integer between 1 and k appears in a at least once.

It is guaranteed that the sum of n over all test cases does not exceed $4 \cdot 10^5$.

Output

For each test case, output a single integer — the minimum number of operations needed to make at least one subarray of a *colorful*.

Standard Input	Standard Output
6 3 2 1 2 1 7 3 2 1 1 3 1 1 2 6 3 1 1 2 2 2 3 6 3 1 2 2 2 2 3 10 5	0 1 2 3 4 5

5 1 3 1 1 2 2 4 1 3 9 4 1 2 3 3 3 3 3 2 4	

Note

In the first test case, since the subarrays $[a_1, a_2] = [1, 2]$ and $[a_2, a_3] = [2, 1]$ are already *colorful*, we do not need to perform any operations. Thus, the answer is 0.

In the second test case, we can swap a_1 and a_2 to obtain $[1, \underline{2, 1, 3}, 1, 1, 2]$, which has a *colorful* subarray $[a_2, a_3, a_4] = [2, 1, 3]$. And the given array initially does not have any *colorful* subarrays, so the answer is 1.