A. Two Friends

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

Monocarp wants to throw a party. He has n friends, and he wants to have at least 2 of them at his party.

The i-th friend's best friend is p_i. All p_i are distinct, and for every $i \in [1, n]$, $p_i \neq i$.

Monocarp can send invitations to friends. The i-th friend comes to the party if **both the i-th friend and the p_i-th friend** receive an invitation (note that the p_i-th friend doesn't have to actually come to the party). Each invitation is sent to exactly one of the friends.

For example, if $p = [3, 1, 2, 5, 4]$, and Monocarp sends invitations to the friends $[1, 2, 4, 5]$, then the friends $[2, 4, 5]$ will come to the party. The friend 1 won't come since his best friend didn't receive an invitation; the friend 3 won't come since he didn't receive an invitation.

Calculate the minimum number of invitations Monocarp has to send so that **at least 2** friends come to the party.

Input

The first line contains one integer t ($1 \le t \le 5000$) — the number of test cases.

Each test case consists of two lines:

- the first line contains one integer n ($2 \le n \le 50$) — the number of friends;
- the second line contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$; $p_i \neq i$; all p_i are distinct).

Output

Print one integer — the minimum number of invitations Monocarp has to send.

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

Note

In the first testcase, Monocarp can send invitations to friends 4 and 5. Both of them will come to the party since they are each other's best friends, and both of them have invitations.

In the second testcase, Monocarp can send invitations to friends $1, 2$ and 3, for example. Then friends 1 and 2 will attend: friend 1 and his best friend 2 have invitations, friend 2 and his best friend 3 have invitations. Friend 3 won't attend since his friend 4 doesn't have an invitation. It's impossible to send invitations to fewer than 3 friends in such a way that at least 2 come.

In the third testcase, Monocarp can send invitations to both friends 1 and 2, and both of them will attend.

,

B. Shifts and Sorting

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

Let's define a *cyclic shift* of some string s as a transformation from $s_1 s_2 \dots s_{n-1} s_n$ into $s_n s_1 s_2 \dots s_{n-1}$. In other words, you take one last character s_n and place it before the first character while moving all other characters to the right.

You are given a binary string s (a string consisting of only 0-s and/or 1-s).

In one operation, you can choose any substring $s_l s_{l+1} \dots s_r$ ($1 \le l < r \le |s|$) and cyclically shift it. The *cost* of such operation is equal to $r - l + 1$ (or the length of the chosen substring).

You can perform the given operation any number of times. What is the minimum **total** cost to make s sorted in non-descending order?

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 a binary string s ($2 \le |s| \le 2 \cdot 10^5$; $s_i \in$ {0, 1}) — the string you need to sort.

Additional constraint on the input: the sum of lengths of strings over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print the single integer — the minimum total cost to make string sorted using operation above any number of times.

Standard Input	Standard Output
5 10 0000 11000 101011 01101001	2 0 9 5 11

Note

In the first test case, you can choose the whole string and perform a cyclic shift: 10 $\rightarrow$ 01. The length of the substring is 2, so the cost is 2.

In the second test case, the string is already sorted, so you don't need to perform any operations.

In the third test case, one of the optimal strategies is the next:

1. choose substring $[1, 3]$: 11000 $\rightarrow$ 01100;
2. choose substring $[2, 4]$: 01100 $\rightarrow$ 00110;
3. choose substring $[3, 5]$: 00110 $\rightarrow$ 00011.

The total cost is $3 + 3 + 3 = 9$.

,

C. Minimizing the Sum

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

You are given an integer array a of length n.

You can perform the following operation: choose an element of the array and replace it with any of its neighbor's value.

For example, if $a = [3, 1, 2]$, you can get one of the arrays $[3, 3, 2]$, $[3, 2, 2]$ and $[1, 1, 2]$ using one operation, but not $[2, 1, 2]$ or $[3, 4, 2]$.

Your task is to calculate the minimum possible total sum of the array if you can perform the aforementioned operation at most k times.

Input

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

The first line of each test case contains two integers n and k ($1 \le n \le 3 \cdot 10^5$; $0 \le k \le 10$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

Additional constraint on the input: the sum of n over all test cases doesn't exceed $3 \cdot 10^5$.

Output

For each test case, print a single integer — the minimum possible total sum of the array if you can perform the aforementioned operation at most k times.

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

Note

In the first example, one of the possible sequences of operations is the following: $[3, 1, 2] \to [1, 1, 2]$.

In the second example, you do not need to apply the operation.

In the third example, one of the possible sequences of operations is the following:
$[2, 2, 1, 3] \to [2, 1, 1, 3] \to [2, 1, 1, 1]$.

In the fourth example, one of the possible sequences of operations is the following:
$[4, 1, 2, 2, 4, 3] \to [1, 1, 2, 2, 4, 3] \to [1, 1, 1, 2, 4, 3] \to [1, 1, 1, 2, 2, 3]$.

,

D. Shop Game

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

Alice and Bob are playing a game in the shop. There are n items in the shop; each item has two parameters: a_i (item price for Alice) and b_i (item price for Bob).

Alice wants to choose a subset (possibly empty) of items and buy them. After that, Bob does the following:

- if Alice bought less than k items, Bob can take all of them for free;
- otherwise, he will take k items for free that Alice bought (Bob chooses which k items it will be), and for the rest of the chosen items, Bob will buy them from Alice and pay b_i for the i-th item.

Alice's profit is equal to $\sum\limits_{i \in S} b_i - \sum\limits_{j \in T} a_j$, where S is the set of items Bob buys from Alice, and T is the set of items Alice buys from the shop. In other words, Alice's profit is the difference between the amount Bob pays her and the amount she spends buying the items.

Alice wants to maximize her profit, Bob wants to minimize Alice's profit. Your task is to calculate Alice's profit if both Alice and Bob act optimally.

Input

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

The first line of each test case contains two integers n and k ($1 \le n \le 2 \cdot 10^5$; $0 \le k \le n$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$).

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^9$).

Additional constraint on the input: the sum of n over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print a single integer — Alice's profit if both Alice and Bob act optimally.

Standard Input	Standard Output
4 2 0 2 1 1 2 4 1 1 2 1 4 3 3 2 3 4 2 2 1 1 1 4 2 3 2 6 2 1 3 4 9 1 3 7 6 8 10 6 8	1 1 0 7

Note

In the first test case, Alice should buy the 2-nd item and sell it to Bob, so her profit is $2 - 1 = 1$.

In the second test case, Alice should buy the 1-st, the 2-nd and the 3-rd item; then Bob takes the 1-st item for free and pays for the 2-nd and the 3-rd item. Alice's profit is $(3 + 2) - (1 + 2 + 1) = 1$. Bob could take 2-nd item for free instead; this does not change Alice's profit. Bob won't take the 3-rd item for free, since this would lead to a profit of 2.

,

E. Unique Array

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

You are given an integer array a of length n. A subarray of a is one of its contiguous subsequences (i. e. an array $[a_l, a_{l+1}, \ldots, a_r]$ for some integers l and r such that $1 \le l < r \le n$). Let's call a subarray *unique* if there is an integer that occurs exactly once in the subarray.

You can perform the following operation any number of times (possibly zero): choose an element of the array and replace it with any integer.

Your task is to calculate the minimum number of aforementioned operation in order for all the subarrays of the array a to be unique.

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 3 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$).

Additional constraint on the input: the sum of n over all test cases doesn't exceed $3 \cdot 10^5$.

Output

For each test case, print a single integer — the minimum number of aforementioned operation in order for all the subarrays of the array a to be unique.

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

Note

In the second test case, you can replace the 1-st and the 3-rd element, for example, like this: $[3, 4, 1, 4]$.

In the third test case, you can replace the 4-th element, for example, like this: $[3, 1, 2, 3, 2]$.

,

F. Card Pairing

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

There is a deck of n cards, each card has one of k types. You are given the sequence $a_1, a_2, \ldots, a_n$ denoting the types of cards in the deck from top to bottom. Both n and k are even numbers.

You play a game with these cards. First, you draw k topmost cards from the deck. Then, the following happens each turn of the game:

- you choose **exactly** two cards from your hand and play them. If these cards have the same type, you earn a coin;
- then, if the deck is not empty, you draw **exactly** two top cards from it;
- then, if both your hand and your deck are empty, the game ends. Otherwise, the new turn begins.

You have to calculate the maximum number of coins you can earn during the game.

Input

The first line of the input contains two integers n and k ($2 \le k \le n \le 1000$, both n and k are even).

The second line of the input contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le k$).

Output

Print one integer — the maximum number of coins you can earn.

Standard Input	Standard Output
4 2 1 2 1 2	0
8 2 2 1 2 2 1 2 1 2	1
4 4 1 2 1 2	2
6 4 3 2 3 1 2 1	3
6 4 3 2 3 3 2 1	2
18 6 5 6 1 1 6 5 4 1 5 1 1 6 2 6 2 2 6 3	6
8 4 1 2 3 4 4 3 1 2	2
8 4 1 2 3 4 4 3 3 2	3