Problem

As the football coach at your local school, you have been tasked with picking a team of exactly **P** students to represent your school. There are **N** students for you to pick from. The i-th student has a *skill rating* $\mathbf{S_i}$, which is a positive integer indicating how skilled they are.

You have decided that a team is *fair* if it has exactly **P** students on it and they all have the same skill rating. That way, everyone plays as a team. Initially, it might not be possible to pick a fair team, so you will give some of the students one-on-one coaching. It takes one hour of coaching to increase the skill rating of any student by 1.

The competition season is starting very soon (in fact, the first match has already started!), so you'd like to find the minimum number of hours of coaching you need to give before you are able to pick a fair team.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line containing the two integers **N** and **P**, the number of students and the number of students you need to pick, respectively. Then, another line follows containing **N** integers $\mathbf{S_i}$; the i-th of these is the skill of the i-th student.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the minimum number of hours of coaching needed, before you can pick a fair team of **P** students.

Limits

Time limit: 15 seconds per test set.
Memory limit: 1 GB.
$1 \leq \mathbf{T} \leq 100$.
$1 \leq \mathbf{S_i} \leq 10000$, for all i.
$2 \leq \mathbf{P} \leq \mathbf{N}$.

Test set 1 (Visible)

$2 \leq \mathbf{N} \leq 1000$.

Test set 2 (Hidden)

$2 \leq \mathbf{N} \leq 10^5$.

Sample

Input	Output
3 4 3 3 1 9 100 6 2 5 5 1 2 3 4 5 5 7 7 1 7 7	Case #1: 14 Case #2: 0 Case #3: 6

In Sample Case #1, you can spend a total of 6 hours training the first student and 8 hours training the second one. This gives the first, second and third students a skill level of 9. This is the minimum time you can spend, so the answer is 14.

In Sample Case #2, you can already pick a fair team (the first and second student) without having to do any coaching, so the answer is 0.

In Sample Case #3, **P** = **N**, so every student will be on your team. You have to spend 6 hours training the third student, so that they have a skill of 7, like everyone else. This is the minimum time you can spend, so the answer is 6.

Analysis

To make a fair team, we have to train the members of the team to the same skill level as the most skillful member of the team.
For any **P** students we pick, the time required to make a fair team is = $\Sigma(\max(\mathbf{S}_i, \mathbf{S}_{i+1} \ldots \mathbf{S}_P) - \mathbf{S}_i)$, for all students i = 1 to **P** in the team. Our goal is to minimize this value.
One possible approach could be to go through all possible subsets of **P** students, from the given **N** students. But there exists ${}^{\mathbf{N}}C_{\mathbf{P}}$ such subsets(here symbol C denotes Combination). Number of such subsets will be in the order of Factorial(**N**) and so enumerating through all of them will not fit within the time limit.

Test set 1 (Visible)

We can start with the observation that once we fix the student with highest skill level x, to minimize our goal we should always choose students with skills as high as possible, but less than or equal to x. Hence we can sort students on the basis of skill level in decreasing order, and iterate over each student assuming they would have the highest skill level in the team. Say, this student is at position i in the sorted sequence; the team would be formed of students on positions i, i + 1, ..., i + **P** - 1 (i.e. a

contiguous subarray of size **P**).
For each subarray of size **P** in the sorted array, we can calculate the training time required to make a fair team using the aforementioned formula. There are **N** - **P** + 1 subarrays of size **P**, and the complexity of calculating training time of subarray size **P** is O(**P**). So the overall complexity of this approach is O(**N** × **P**), which will be sufficient for test set 1.

Test set 2 (Hidden)

We need to go through all of the subarrays once, but can we calculate the training time of a subarray faster?
Let us assume the array is sorted in decreasing order. The training time formula for a subarray starting at position i is
= Σ(**S**[i] - **S**[j]) where j = i to i + **P** -1
= **P** × **S**[i] - Σ(**S**[j]) where j = i to i + **P** - 1
As we always need sum of a contagious subarray, we can pre compute the prefix sum of the whole array in advance, and get the sum of any subarray in O(1) time, which makes the calculation of training time O(1).
So, the overall complexity of this approach is O(**N**).