Codeforces Round #570 (Div. 3)

A. Nearest Interesting Number

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp knows that if the sum of the digits of a number is divisible by 3, then the number itself is divisible by 3. He assumes that the numbers, the sum of the digits of which is divisible by 4, are also somewhat interesting. Thus, he considers a positive integer n interesting if its sum of digits is divisible by 4.

Help Polycarp find the nearest larger or equal interesting number for the given number a. That is, find the interesting number n such that $n \geq a$ and n is minimal.

Input
The only line in the input contains an integer a ($1 \leq a \leq 1000$).

Output
Print the nearest greater or equal interesting number for the given number a. In other words, print the interesting number n such that $n \geq a$ and n is minimal.

Examples

input
432
output
435

input
99
output
103

input
237
output
237

input
42
output
44

B. Equalize Prices

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

There are n products in the shop. The price of the i-th product is a_i. The owner of the shop wants to equalize the prices of all products. However, he wants to change prices smoothly.

In fact, the owner of the shop can change the price of some product i in such a way that the difference between the old price of this product a_i and the new price b_i is at most k. In other words, the condition $|a_i - b_i| \leq k$ should be satisfied ($|x|$ is the absolute value of x).

He can change the price for each product **not more than once**. Note that he can leave the old prices for some products. The new price b_i of each product i should be positive (i.e. $b_i > 0$ should be satisfied for all i from 1 to n).

Your task is to find out the **maximum** possible **equal** price B of **all** productts with the restriction that for all products the condiion $|a_i - B| \leq k$ should be satisfied (where a_i is the old price of the product and B is the same new price of all products) or report that it is impossible to find such price B.

Note that the chosen price B should be integer.

You should answer q independent queries.

Input
The first line of the input contains one integer q ($1 \le q \le 100$) — the number of queries. Each query is presented by two lines.

The first line of the query contains two integers n and k ($1 \le n \le 100, 1 \le k \le 10^8$) — the number of products and the value k. The second line of the query contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^8$), where a_i is the price of the i-th product.

Output
Print q integers, where the i-th integer is the answer B on the i-th query.

If it is impossible to equalize prices of **all** given products with restriction that for all products the condition $|a_i - B| \le k$ should be satisfied (where a_i is the old price of the product and B is the new equal price of all products), print -1. Otherwise print the **maximum** possible equal price of **all** products.

Example

input
4 5 1 1 1 2 3 1 4 2 6 4 8 5 2 2 1 6 3 5 5 2 5
output
2 6 -1 7

Note
In the first example query you can choose the price $B = 2$. It is easy to see that the difference between each old price and each new price $B = 2$ is no more than 1.

In the second example query you can choose the price $B = 6$ and then all the differences between old and new price $B = 6$ will be no more than 2.

In the third example query you cannot choose any suitable price B. For any value B at least one condition out of two will be violated: $|1 - B| \le 2$, $|6 - B| \le 2$.

In the fourth example query all values B between 1 and 7 are valid. But the maximum is 7, so it's the answer.

C. Computer Game

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Vova is playing a computer game. There are in total n turns in the game and Vova really wants to play all of them. The initial charge of his laptop battery (i.e. the charge before the start of the game) is k.

During each turn Vova can choose what to do:

- If the current charge of his laptop battery is strictly greater than a, Vova can *just play*, and then the charge of his laptop battery will decrease by a;
- if the current charge of his laptop battery is strictly greater than b ($b < a$), Vova can *play and charge* his laptop, and then the charge of his laptop battery will decrease by b;
- if the current charge of his laptop battery is less than or equal to a and b at the same time then Vova cannot do anything and loses the game.

Regardless of Vova's turns the charge of the laptop battery is always decreases.

Vova wants to complete the game (Vova can complete the game if after each of n turns the charge of the laptop battery is **strictly greater than** 0). Vova has to play **exactly** n **turns**. Among all possible ways to complete the game, Vova wants to choose the one where the number of turns when he *just plays* (**first type turn**) is the **maximum** possible. It is possible that Vova cannot complete the game at all.

Your task is to find out the **maximum** possible number of turns Vova can *just play* (make the **first type turn**) or report that Vova cannot complete the game.

You have to answer q independent queries.

Input

The first line of the input contains one integer q ($1 \le q \le 10^5$) — the number of queries. Each query is presented by a single line.

The only line of the query contains four integers k, n, a and b ($1 \le k, n \le 10^9, 1 \le b < a \le 10^9$) — the initial charge of Vova's laptop battery, the number of turns in the game and values a and b, correspondingly.

Output
For each query print one integer: -1 if Vova cannot complete the game or the **maximum** number of turns Vova can *just play* (make the **first type turn**) otherwise.

Example

input
6 15 5 3 2 15 5 4 3 15 5 2 1 15 5 5 1 16 7 5 2 20 5 7 3
output
4 -1 5 2 0 1

Note
In the first example query Vova can *just play* 4 turns and spend 12 units of charge and then one turn *play and charge* and spend 2 more units. So the remaining charge of the battery will be 1.

In the second example query Vova cannot complete the game because even if he will *play and charge* the battery during each turn then the charge of the laptop battery will be 0 after the last turn.

D. Candy Box (easy version)

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem is actually a subproblem of problem G from the same contest.

There are n candies in a candy box. The type of the i-th candy is a_i ($1 \le a_i \le n$).

You have to prepare a gift using some of these candies with the following restriction: the numbers of candies of each type presented in a gift should be all distinct (i. e. for example, a gift having two candies of type 1 and two candies of type 2 is bad).

It is possible that multiple types of candies are completely absent from the gift. It is also possible that **not all candies** of some types will be taken to a gift.

Your task is to find out the **maximum** possible size of the single gift you can prepare using the candies you have.

You have to answer q independent queries.

If you are Python programmer, consider using PyPy instead of Python when you submit your code.

Input
The first line of the input contains one integer q ($1 \le q \le 2 \cdot 10^5$) — the number of queries. Each query is represented by two lines.

The first line of each query contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of candies.

The second line of each query contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le n$), where a_i is the type of the i-th candy in the box.

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

Output
For each query print one integer — the **maximum** possible size of the single gift you can compose using candies you got in this query with the restriction described in the problem statement.

Example

input
3 8 1 4 8 4 5 6 3 8 16 2 1 3 3 4 3 4 4 1 3 2 2 2 4 1 1 9 2 2 4 4 4 7 7 7 7

output
3 10 9

Note
In the first query, you can prepare a gift with two candies of type 8 and one candy of type 5, totalling to 3 candies.

Note that this is not the only possible solution — taking two candies of type 4 and one candy of type 6 is also valid.

E. Subsequences (easy version)

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The only difference between the easy and the hard versions is constraints.

A subsequence is a string that can be derived from another string by deleting some or no symbols without changing the order of the remaining symbols. Characters to be deleted are not required to go successively, there can be any gaps between them. For example, for the string "abaca" the following strings are subsequences: "abaca", "aba", "aaa", "a" and "" (empty string). But the following strings are not subsequences: "aabaca", "cb" and "bcaa".

You are given a string s consisting of n lowercase Latin letters.

In one move you can take **any** subsequence t of the given string and add it to the set S. The set S can't contain duplicates. This move costs $n - |t|$, where $|t|$ is the length of the added subsequence (i.e. the price equals to the number of the deleted characters).

Your task is to find out the minimum possible total cost to obtain a set S of size k or report that it is impossible to do so.

Input
The first line of the input contains two integers n and k ($1 \le n, k \le 100$) — the length of the string and the size of the set, correspondingly.

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

Output
Print one integer — if it is impossible to obtain the set S of size k, print -1. Otherwise, print the minimum possible total cost to do it.

Examples

input
4 5 asdf
output
4

input
5 6 aaaaa
output
15

input
5 7 aaaaa
output
-1

input
10 100 ajihiushda
output
233

Note
In the first example we can generate S = { "asdf", "asd", "adf", "asf", "sdf" }. The cost of the first element in S is 0 and the cost of the others is 1. So the total cost of S is 4.

F. Topforces Strikes Back

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

One important contest will take place on the most famous programming platform (Topforces) very soon!

The authors have a pool of n problems and should choose **at most three** of them into this contest. The prettiness of the i-th problem is a_i. The authors have to compose the most pretty contest (in other words, the cumulative prettinesses of chosen problems should be **maximum possible**).

But there is one important thing in the contest preparation: because of some superstitions of authors, the prettinesses of problems cannot divide each other. In other words, if the prettinesses of chosen problems are x, y, z, then x should be divisible by neither y, nor z, y should be divisible by neither x, nor z and z should be divisible by neither x, nor y. If the prettinesses of chosen problems are x and y then neither x should be divisible by y nor y should be divisible by x. Any contest composed from one problem is considered good.

Your task is to find out the maximum possible total prettiness of the contest composed of **at most three** problems from the given pool.

You have to answer q independent queries.

If you are Python programmer, consider using PyPy instead of Python when you submit your code.

Input

The first line of the input contains one integer q ($1 \le q \le 2 \cdot 10^5$) — the number of queries.

The first line of the query contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of problems.

The second line of the query contains n integers $a_1, a_2, \ldots, a_n$ ($2 \le a_i \le 2 \cdot 10^5$), where a_i is the prettiness of the i-th problem.

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

Output

For each query print one integer — the maximum possible cumulative prettiness of the contest composed of **at most three** problems from the given pool of problems in the query.

Example

input
3 4 5 6 15 30 4 10 6 30 15 3 3 4 6
output
30 31 10

G. Candy Box (hard version)

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem is a version of problem D from the same contest with some additional constraints and tasks.

There are n candies in a candy box. The type of the i-th candy is a_i ($1 \le a_i \le n$).

You have to prepare a gift using some of these candies with the following restriction: the numbers of candies of each type presented in a gift should be all distinct (i. e. for example, a gift having two candies of type 1 and two candies of type 2 is bad).

It is possible that multiple types of candies are completely absent from the gift. It is also possible that **not all candies** of some types will be taken to a gift.

You really like some of the candies and don't want to include them into the gift, but you want to eat them yourself instead. For each candy, a number f_i is given, which is equal to 0 if you really want to keep i-th candy for yourself, or 1 if you don't mind including it into your gift. It is possible that two candies of the same type have different values of f_i.

You want your gift to be as large as possible, but you don't want to include too many of the candies you want to eat into the gift. So, you want to calculate the maximum possible number of candies that can be included into a gift, and among all ways to choose maximum number of candies, you want to maximize the number of candies having $f_i = 1$ in your gift.

You have to answer q independent queries.

If you are Python programmer, consider using PyPy instead of Python when you submit your code.

Input
The first line of the input contains one integer q ($1 \le q \le 2 \cdot 10^5$) — the number of queries.

The first line of each query contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of candies.

Then n lines follow, each containing two integers a_i and f_i ($1 \le a_i \le n$, $0 \le f_i \le 1$), where a_i is the type of the i-th candy, and f_i denotes whether you want to keep the i-th candy for yourself (0 if you want to keep it, 1 if you don't mind giving it away).

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

Output
For each query print two integers:

- the maximum number of candies in a gift you can compose, according to the constraints in the statement;
- the maximum number of candies having $f_i = 1$ in a gift you can compose that contains the maximum possible number of candies.

Example

input
3 8 1 0 4 1 2 0 4 1 5 1 6 1 3 0 2 0 4 1 1 1 1 2 1 2 1 9 2 0 2 0 4 1 4 1 4 1 7 0 7 1 7 0 7 1
output
3 3 3 3 9 5

Note
In the first query, you can include two candies of type 4 and one candy of type 5. All of them have $f_i = 1$ and you don't mind giving them away as part of the gift.

H. Subsequences (hard version)

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The only difference between the easy and the hard versions is constraints.

A subsequence is a string that can be derived from another string by deleting some or no symbols without changing the order of the remaining symbols. Characters to be deleted are not required to go successively, there can be any gaps between them. For example, for the string "abaca" the following strings are subsequences: "abaca", "aba", "aaa", "a" and "" (empty string). But the following strings are not subsequences: "aabaca", "cb" and "bcaa".

You are given a string s consisting of n lowercase Latin letters.

In one move you can take **any** subsequence t of the given string and add it to the set S. The set S can't contain duplicates. This move costs $n - |t|$, where $|t|$ is the length of the added subsequence (i.e. the price equals to the number of the deleted characters).

Your task is to find out the minimum possible total cost to obtain a set S of size k or report that it is impossible to do so.

Input
The first line of the input contains two integers n and k ($1 \le n \le 100, 1 \le k \le 10^{12}$) — the length of the string and the size of the set, correspondingly.

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

Output

Print one integer — if it is impossible to obtain the set S of size k, print -1. Otherwise, print the minimum possible total cost to do it.

Examples

input
4 5 asdf
output
4

input
5 6 aaaaa
output
15

input
5 7 aaaaa
output
-1

input
10 100 ajihiushda
output
233

Note

In the first example we can generate S = { "asdf", "asd", "adf", "asf", "sdf" }. The cost of the first element in S is 0 and the cost of the others is 1. So the total cost of S is 4.