Educational Codeforces Round 96 (Rated for Div. 2)

A. Number of Apartments

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

Recently a new building with a new layout was constructed in Monocarp's hometown. According to this new layout, the building consists of three types of apartments: three-room, five-room, and seven-room apartments. It's also known that each room of each apartment has exactly one window. In other words, a three-room apartment has three windows, a five-room — five windows, and a seven-room — seven windows.

Monocarp went around the building and counted n windows. Now he is wondering, how many apartments of each type the building may have.

Unfortunately, Monocarp only recently has learned to count, so he is asking you to help him to calculate the possible quantities of three-room, five-room, and seven-room apartments in the building that has n windows. If there are multiple answers, you can print any of them.

Here are some examples:

- if Monocarp has counted 30 windows, there could have been 2 three-room apartments, 2 five-room apartments and 2 seven-room apartments, since $2 \cdot 3 + 2 \cdot 5 + 2 \cdot 7 = 30$;
- if Monocarp has counted 67 windows, there could have been 7 three-room apartments, 5 five-room apartments and 3 seven-room apartments, since $7 \cdot 3 + 5 \cdot 5 + 3 \cdot 7 = 67$;
- if Monocarp has counted 4 windows, he should have mistaken since no building with the aforementioned layout can have 4 windows.

Input

Th first line contains one integer t ($1 \le t \le 1000$) — the number of test cases.

The only line of each test case contains one integer n ($1 \le n \le 1000$) — the number of windows in the building.

Output

For each test case, if a building with the new layout and the given number of windows just can't exist, print -1.

Otherwise, print three non-negative integers — the possible number of three-room, five-room, and seven-room apartments. If there are multiple answers, print any of them.

Example

input
4 30 67 4 14
output
2 2 2 7 5 3 -1 0 0 2

B. Barrels

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

You have n barrels lined up in a row, numbered from left to right from one. Initially, the i-th barrel contains a_i liters of water.

You can pour water from one barrel to another. In one act of pouring, you can choose two different barrels x and y (the x-th barrel shouldn't be empty) and pour any possible amount of water from barrel x to barrel y (possibly, all water). You may assume that barrels have infinite capacity, so you can pour any amount of water in each of them.

Calculate the maximum possible difference between the maximum and the minimum amount of water in the barrels, if you can pour water **at most** k times.

Some examples:

- if you have four barrels, each containing 5 liters of water, and $k = 1$, you may pour 5 liters from the second barrel into the fourth, so the amounts of water in the barrels are $[5, 0, 5, 10]$, and the difference between the maximum and the minimum is 10;
- if all barrels are empty, you can't make any operation, so the difference between the maximum and the minimum amount is still 0.

Input
The first line contains one integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains two integers n and k ($1 \le k < n \le 2 \cdot 10^5$) — the number of barrels and the number of pourings you can make.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($0 \le a_i \le 10^9$), where a_i is the initial amount of water the i-th barrel has.

It's guaranteed that the total sum of n over test cases doesn't exceed $2 \cdot 10^5$.

Output
For each test case, print the maximum possible difference between the maximum and the minimum amount of water in the barrels, if you can pour water **at most** k times.

Example

input
2 4 1 5 5 5 5 3 2 0 0 0
output
10 0

C. Numbers on Whiteboard

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

Numbers $1, 2, 3, \dots n$ (each integer from 1 to n once) are written on a board. In one operation you can erase any two numbers a and b from the board and write one integer $\frac{a+b}{2}$ *rounded up* instead.

You should perform the given operation $n - 1$ times and make the resulting number that will be left on the board as small as possible.

For example, if $n = 4$, the following course of action is optimal:

1. choose $a = 4$ and $b = 2$, so the new number is 3, and the whiteboard contains $[1, 3, 3]$;
2. choose $a = 3$ and $b = 3$, so the new number is 3, and the whiteboard contains $[1, 3]$;
3. choose $a = 1$ and $b = 3$, so the new number is 2, and the whiteboard contains $[2]$.

It's easy to see that after $n - 1$ operations, there will be left only one number. Your goal is to minimize it.

Input
The first line contains one integer t ($1 \le t \le 1000$) — the number of test cases.

The only line of each test case contains one integer n ($2 \le n \le 2 \cdot 10^5$) — the number of integers written on the board initially.

It's guaranteed that the total sum of n over test cases doesn't exceed $2 \cdot 10^5$.

Output
For each test case, in the first line, print the minimum possible number left on the board after $n - 1$ operations. Each of the next $n - 1$ lines should contain two integers — numbers a and b chosen and erased in each operation.

Example

input
1 4
output
2 2 4 3 3 3 1

D. String Deletion

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

You have a string s consisting of n characters. Each character is either 0 or 1.

You can perform operations on the string. Each operation consists of two steps:

1. select an integer i from 1 to the length of the string s, then delete the character s_i (the string length gets reduced by 1, the indices of characters to the right of the deleted one also get reduced by 1);
2. if the string s is not empty, delete the maximum length prefix consisting of the same characters (the indices of the remaining characters and the string length get reduced by the length of the deleted prefix).

Note that both steps are mandatory in each operation, and their order cannot be changed.

For example, if you have a string $s = 111010$, the first operation can be one of the following:

1. select $i = 1$: we'll get $111010 \rightarrow 11010 \rightarrow 010$;
2. select $i = 2$: we'll get $111010 \rightarrow 11010 \rightarrow 010$;
3. select $i = 3$: we'll get $111010 \rightarrow 11010 \rightarrow 010$;
4. select $i = 4$: we'll get $111010 \rightarrow 11110 \rightarrow 0$;
5. select $i = 5$: we'll get $111010 \rightarrow 11100 \rightarrow 00$;
6. select $i = 6$: we'll get $111010 \rightarrow 11101 \rightarrow 01$.

You finish performing operations when the string s becomes empty. What is the maximum number of operations you can perform?

Input

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

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

The second line contains string s of n characters. Each character is either 0 or 1.

It's guaranteed that the total sum of n over test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print a single integer — the maximum number of operations you can perform.

Example

input
5 6 111010 1 0 1 1 2 11 6 101010
output
3 1 1 1 3

Note

In the first test case, you can, for example, select $i = 2$ and get string 010 after the first operation. After that, you can select $i = 3$ and get string 1. Finally, you can only select $i = 1$ and get empty string.

E. String Reversal

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

You are given a string s. You have to reverse it — that is, the first letter should become equal to the last letter before the reversal, the second letter should become equal to the second-to-last letter before the reversal — and so on. For example, if your goal is to reverse the string "abddea", you should get the string "aeddba". To accomplish your goal, you can swap the **neighboring elements of the string**.

Your task is to calculate the minimum number of swaps you have to perform to reverse the given string.

Input
The first line contains one integer n ($2 \le n \le 200\,000$) — the length of s.

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

Output
Print one integer — the minimum number of swaps of neighboring elements you have to perform to reverse the string.

Examples

input
5 aaaza
output
2

input
6 cbaabc
output
0

input
9 icpcsguru
output
30

Note
In the first example, you have to swap the third and the fourth elements, so the string becomes "aazaa". Then you have to swap the second and the third elements, so the string becomes "azaaa". So, it is possible to reverse the string in two swaps.

Since the string in the second example is a palindrome, you don't have to do anything to reverse it.

F. Realistic Gameplay

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

Recently you've discovered a new shooter. They say it has realistic game mechanics.

Your character has a gun with magazine size equal to k and should exterminate n waves of monsters. The i-th wave consists of a_i monsters and happens from the l_i-th moment of time up to the r_i-th moments of time. All a_i monsters spawn at moment l_i and you have to exterminate all of them before the moment r_i ends (you can kill monsters right at moment r_i). For every two consecutive waves, the second wave starts not earlier than the first wave ends (though the second wave can start at the same moment when the first wave ends) — formally, the condition $r_i \le l_{i+1}$ holds. Take a look at the notes for the examples to understand the process better.

You are confident in yours and your character's skills so you can assume that aiming and shooting are instant and you need exactly one bullet to kill one monster. But reloading takes exactly 1 unit of time.

One of the realistic mechanics is a mechanic of reloading: when you reload you throw away the old magazine with all remaining bullets in it. That's why constant reloads may cost you excessive amounts of spent bullets.

You've taken a liking to this mechanic so now you are wondering: what is the minimum possible number of bullets you need to spend (both used and thrown) to exterminate all waves.

Note that you don't throw the remaining bullets away after eradicating all monsters, and you start with a full magazine.

Input
The first line contains two integers n and k ($1 \le n \le 2000$; $1 \le k \le 10^9$) — the number of waves and magazine size.

The next n lines contain descriptions of waves. The i-th line contains three integers l_i, r_i and a_i ($1 \le l_i \le r_i \le 10^9$; $1 \le a_i \le 10^9$) — the period of time when the i-th wave happens and the number of monsters in it.

It's guaranteed that waves don't overlap (but may touch) and are given in the order they occur, i. e. $r_i \le l_{i+1}$.

Output
If there is no way to clear all waves, print -1. Otherwise, print the minimum possible number of bullets you need to spend (both used and thrown) to clear all waves.

Examples

input
2 3 2 3 6 3 4 3
output
9

input
2 5 3 7 11 10 12 15
output
30

input
5 42 42 42 42 42 43 42 43 44 42 44 45 42 45 45 1
output
-1

input
1 10 100 111 1
output
1

Note

In the first example:

- At the moment 2, the first wave occurs and 6 monsters spawn. You kill 3 monsters and start reloading.
- At the moment 3, the second wave occurs and 3 more monsters spawn. You kill remaining 3 monsters from the first wave and start reloading.
- At the moment 4, you kill remaining 3 monsters from the second wave.

In total, you'll spend 9 bullets.
In the second example:

- At moment 3, the first wave occurs and 11 monsters spawn. You kill 5 monsters and start reloading.
- At moment 4, you kill 5 more monsters and start reloading.
- At moment 5, you kill the last monster and start reloading throwing away old magazine with 4 bullets.
- At moment 10, the second wave occurs and 15 monsters spawn. You kill 5 monsters and start reloading.
- At moment 11, you kill 5 more monsters and start reloading.
- At moment 12, you kill last 5 monsters.

In total, you'll spend 30 bullets.

G. Yet Another DAG Problem

time limit per test: 2 seconds
memory limit per test: 1024 megabytes
input: standard input
output: standard output

You are given a directed acyclic graph (a directed graph that does not contain cycles) of n vertices and m arcs. The i-th arc leads from the vertex x_i to the vertex y_i and has the weight w_i.

Your task is to select an integer a_v for each vertex v, and then write a number b_i on each arcs i such that $b_i = a_{x_i} - a_{y_i}$. You must select the numbers so that:

- all b_i are positive;
- the value of the expression $\sum_{i=1}^{m} w_i b_i$ is the lowest possible.

It can be shown that for any directed acyclic graph with non-negative w_i, such a way to choose numbers exists.

Input

The first line contains two integers n and m ($2 \le n \le 18$; $0 \le m \le \frac{n(n-1)}{2}$).

Then m lines follow, the i-th of them contains three integers x_i, y_i and w_i ($1 \le x_i, y_i \le n$, $1 \le w_i \le 10^5$, $x_i \ne y_i$) — the description of the i-th arc.

It is guaranteed that the lines describe m arcs of a directed acyclic graph without multiple arcs between the same pair of vertices.

Output

Print n integers a_1, a_2, ..., a_n ($0 \le a_v \le 10^9$), which must be written on the vertices so that all b_i are positive, and the value of the expression $\sum_{i=1}^{m} w_i b_i$ is the lowest possible. If there are several answers, print any of them. It can be shown that the answer always exists, and at least one of the optimal answers satisfies the constraints $0 \le a_v \le 10^9$.

Examples

input
3 2 2 1 4 1 3 2
output
1 2 0

input
5 4 1 2 1 2 3 1 1 3 6 4 5 8
output
43 42 41 1337 1336

input
5 5 1 2 1 2 3 1 3 4 1 1 5 1 5 4 10
output
4 3 2 1 2

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform