Codeforces Round #765 (Div. 2)

A. Ancient Civilization

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

Martian scientists explore Ganymede, one of Jupiter's numerous moons. Recently, they have found ruins of an ancient civilization. The scientists brought to Mars some tablets with writings in a language unknown to science.

They found out that the inhabitants of Ganymede used an alphabet consisting of two letters, and each word was exactly ℓ letters long. So, the scientists decided to write each word of this language as an integer from 0 to $2^\ell - 1$ inclusively. The first letter of the alphabet corresponds to zero bit in this integer, and the second letter corresponds to one bit.

The same word may have various forms in this language. Then, you need to restore the initial form. The process of doing it is described below.

Denote the *distance* between two words as the amount of positions, in which these words differ. For example, the distance between 1001_2 and 1100_2 (in binary) is equal to two, as these words have different letters in the second and the fourth positions, counting from left to right. Further, denote the distance between words x and y as $d(x, y)$.

Let the word have n forms, the i-th of which is described with an integer x_i. All the x_i are not necessarily different, as two various forms of the word can be written the same. Consider some word y. Then, *closeness* of the word y is equal to the sum of distances to each of the word forms, i. e. the sum $d(x_i, y)$ over all $1 \le i \le n$.

The initial form is the word y with minimal possible nearness.

You need to help the scientists and write the program which finds the initial form of the word given all its known forms. Note that the initial form is **not necessarily** equal to any of the n given forms.

Input

The first line contains an integer t ($1 \le t \le 100$) — the number of test cases. The following are descriptions of the test cases.

The first line contains two integers n and ℓ ($1 \le n \le 100$, $1 \le \ell \le 30$) — the amount of word forms, and the number of letters in one word.

The second line contains n integers x_i ($0 \le x_i \le 2^\ell - 1$) — word forms. The integers are not necessarily different.

Output

For each test, print a single integer, the initial form of the word, i. e. such y ($0 \le y \le 2^\ell - 1$) that the sum $d(x_i, y)$ over all $1 \le i \le n$ is minimal possible. Note that y can differ from all the integers x_i.

If there are multiple ways to restore the initial form, print any.

Example

input
7 3 5 18 9 21 3 5 18 18 18 1 1 1 5 30 1 2 3 4 5 6 10 99 35 85 46 78 55 2 1 0 1 8 8 5 16 42 15 83 65 78 42
output
17 18 1 1 39 0 2

Note

In the first test case, the words can be written as $x_1 = 10010_2$, $x_2 = 01001_2$ and $x_3 = 10101_2$ in binary. Let $y = 10001_2$. Then, $d(x_1, y) = 2$ (the difference is in the fourth and the fifth positions), $d(x_2, y) = 2$ (the difference is in the first and the second positions), $d(x_3, y) = 1$ (the difference is in the third position). So, the closeness is $2 + 2 + 1 = 5$. It can be shown that you cannot achieve smaller closeness.

In the second test case, all the forms are equal to 18 (10010_2 in binary), so the initial form is also 18. It's easy to see that closeness is equal to zero in this case.

B. Elementary Particles

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

Martians are actively engaged in interplanetary trade. Olymp City, the Martian city known for its spaceport, has become a place where goods from all the corners of our Galaxy come. To deliver even more freight from faraway planets, Martians need fast spaceships.

A group of scientists conducts experiments to build a fast engine for the new spaceship. In the current experiment, there are n elementary particles, the i-th of them has type a_i.

Denote a subsegment of the particle sequence $(a_1, a_2, \ldots, a_n)$ as a sequence $(a_l, a_{l+1}, \ldots, a_r)$ for some left bound l and right bound r ($1 \le l \le r \le n$). For instance, the sequence $(1\ 4\ 2\ 8\ 5\ 7)$ for $l = 2$ and $r = 4$ has the sequence $(4\ 2\ 8)$ as a subsegment. Two subsegments are considered different if at least one bound of those subsegments differs.

Note that the subsegments can be equal as sequences but still considered different. For example, consider the sequence $(1\ 1\ 1\ 1\ 1)$ and two of its subsegments: one with $l = 1$ and $r = 3$ and another with $l = 2$ and $r = 4$. Both subsegments are equal to $(1\ 1\ 1)$, but still considered different, as their left and right bounds differ.

The scientists want to conduct a reaction to get two different subsegments of the same length. Denote this length k. The resulting pair of subsegments must be *harmonious*, i. e. for **some** i ($1 \le i \le k$) it must be true that the types of particles on the i-th position are the same for these two subsegments. For example, the pair $(1\ 7\ 3)$ and $(4\ 7\ 8)$ is harmonious, as both subsegments have 7 on the second position. The pair $(1\ 2\ 3)$ and $(3\ 1\ 2)$ is not harmonious.

The longer are harmonious subsegments, the more chances for the scientists to design a fast engine. So, they asked you to calculate the maximal possible length of harmonious pair made of different subsegments.

Input

The first line contains an integer t ($1 \le t \le 100$) — the number of test cases. The following are descriptions of the test cases.

The first line contains an integer n ($2 \le n \le 150\,000$) — the amount of elementary particles in the sequence.

The second line contains n integers a_i ($1 \le a_i \le 150\,000$) — types of elementary particles.

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

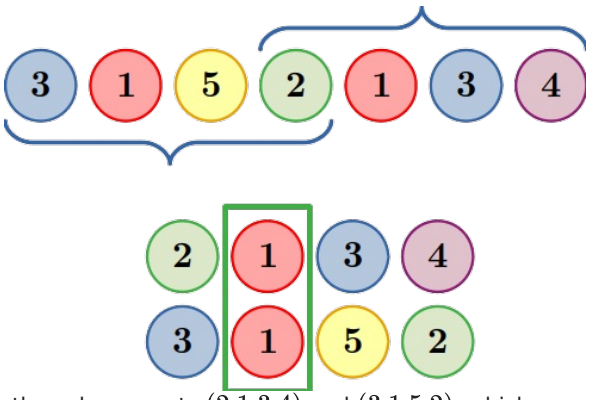
For each test, print a single integer, maximal possible length of harmonious pair made of different subsegments. If such pair does not exist, print -1 instead.

Example

input
4 7 3 1 5 2 1 3 4 6 1 1 1 1 1 1 6 1 4 2 8 5 7 2 15 15
output
4 5 -1 1

Note

The first test case is shown on the picture below:



As you can see from it, you may choose the subsegments $(2\ 1\ 3\ 4)$ and $(3\ 1\ 5\ 2)$, which are a harmonious pair. Their length is equal to 4, so the answer is 4.

In the second test case, you need to take two subsegments: one with $l = 1$ and $r = 5$, and one with $l = 2$ and $r = 6$. It's not hard to observe that these segments are a harmonious pair and considered different even though they are both equal to $(1\ 1\ 1\ 1\ 1)$.

In the third test case, you cannot make a harmonious pair, so the answer is -1.

C. Road Optimization

time limit per test: 3 seconds
memory limit per test: 128 megabytes
input: standard input
output: standard output

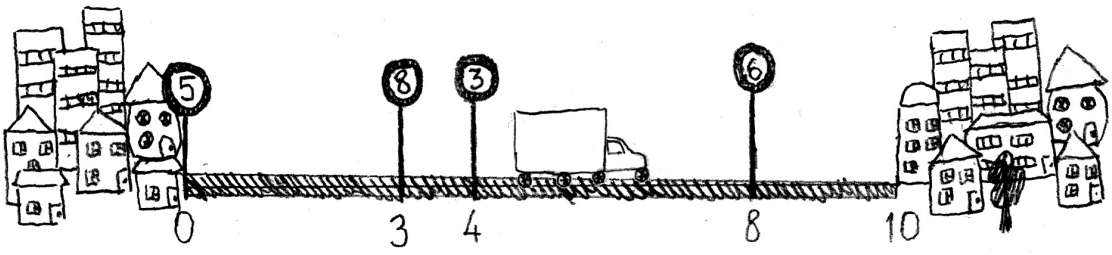
The Government of Mars is not only interested in optimizing space flights, but also wants to improve the road system of the planet.

One of the most important highways of Mars connects Olymp City and Kstolop, the capital of Cydonia. In this problem, we only consider the way from Kstolop to Olymp City, but not the reverse path (i. e. the path from Olymp City to Kstolop).

The road from Kstolop to Olymp City is ℓ kilometers long. Each point of the road has a coordinate x ($0 \le x \le \ell$), which is equal to the distance from Kstolop in kilometers. So, Kstolop is located in the point with coordinate 0, and Olymp City is located in the point with coordinate ℓ.

There are n signs along the road, i-th of which sets a speed limit a_i. This limit means that the next kilometer must be passed in a_i minutes and is active until you encounter the next along the road. There is a road sign at the start of the road (i. e. in the point with coordinate 0), which sets the initial speed limit.

If you know the location of all the signs, it's not hard to calculate how much time it takes to drive from Kstolop to Olymp City. Consider an example:



Here, you need to drive the first three kilometers in five minutes each, then one kilometer in eight minutes, then four kilometers in three minutes each, and finally the last two kilometers must be passed in six minutes each. Total time is $3 \cdot 5 + 1 \cdot 8 + 4 \cdot 3 + 2 \cdot 6 = 47$ minutes.

To optimize the road traffic, the Government of Mars decided to remove no more than k road signs. It cannot remove the sign at the start of the road, otherwise, there will be no limit at the start. By removing these signs, the Government also wants to make the time needed to drive from Kstolop to Olymp City as small as possible.

The largest industrial enterprises are located in Cydonia, so it's the priority task to optimize the road traffic from Olymp City. So, the Government of Mars wants you to remove the signs in the way described above.

Input

The first line contains three integers n, ℓ, k ($1 \le n \le 500$, $1 \le \ell \le 10^5$, $0 \le k \le n - 1$), the amount of signs on the road, the distance between the cities and the maximal number of signs you may remove.

The second line contains n integers d_i ($d_1 = 0$, $d_i < d_{i+1}$, $0 \le d_i \le \ell - 1$) — coordinates of all signs.

The third line contains n integers a_i ($1 \le a_i \le 10^4$) — speed limits.

Output

Print a single integer — minimal possible time to drive from Kstolop to Olymp City in minutes, if you remove no more than k road signs.

Examples

input
4 10 0 0 3 4 8 5 8 3 6
output
47

input
4 10 2 0 3 4 8 5 8 3 6
output
38

Note

In the first example, you cannot remove the signs. So the answer is 47, as it's said in the statements above.

In the second example, you may remove the second and the fourth sign. In this case, you need to drive four kilometers in $4 \cdot 5 = 20$ minutes, and then six kilometers in $6 \cdot 3 = 18$, so the total time is $4 \cdot 5 + 6 \cdot 3 = 38$ minutes.

D. Binary Spiders

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

Binary Spiders are species of spiders that live on Mars. These spiders weave their webs to defend themselves from enemies.

To weave a web, spiders join in pairs. If the first spider in pair has x legs, and the second spider has y legs, then they weave a web with *durability* $x \oplus y$. Here, $\oplus$ means bitwise XOR.

Binary Spiders live in large groups. You observe a group of n spiders, and the i-th spider has a_i legs.

When the group is threatened, some of the spiders become *defenders*. Defenders are chosen in the following way. First, there must be at least two defenders. Second, any pair of defenders must be able to weave a web with durability at least k. Third, there must be as much defenders as possible.

Scientists have researched the behaviour of Binary Spiders for a long time, and now they have a hypothesis that they can always choose the defenders in an optimal way, satisfying the conditions above. You need to verify this hypothesis on your group of spiders. So, you need to understand how many spiders must become defenders. You are not a Binary Spider, so you decided to use a computer to solve this problem.

Input

The first line contains two integers n and k ($2 \le n \le 3 \cdot 10^5$, $0 \le k \le 2^{30} - 1$), the amount of spiders in the group and the minimal allowed durability of a web.

The second line contains n integers a_i ($0 \le a_i \le 2^{30} - 1$) — the number of legs the i-th spider has.

Output

In the first line, print a single integer ℓ ($2 \le \ell \le n$), the maximum possible amount of defenders.

In the second line, print ℓ integers b_i, separated by a single space ($1 \le b_i \le n$) — indices of spiders that will become defenders.

If there exists more than one way to choose the defenders, print any of them.

Unfortunately, it may appear that it's impossible to choose the defenders. In this case, print a single integer -1.

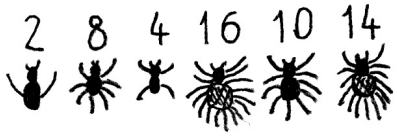
Examples

input
6 8 2 8 4 16 10 14
output
3 1 5 4

input
6 1024 1 2 3 1 4 0
output
-1

Note
Consider the examples above.

In the first example, the group of spiders is illustrated on the picture below:



We choose the two-legged, the ten-legged and the 16-legged spiders. It's not hard to see that each pair may weave a web with enough durability, as $2 \oplus 10 = 8 \geq 8$, $2 \oplus 16 = 18 \geq 8$ and $10 \oplus 16 = 26 \geq 8$.

This is not the only way, as you can also choose, for example, the spiders with indices 3, 4, and 6.

In the second example, no pair of spiders can weave the web with durability 1024 or more, so the answer is -1.

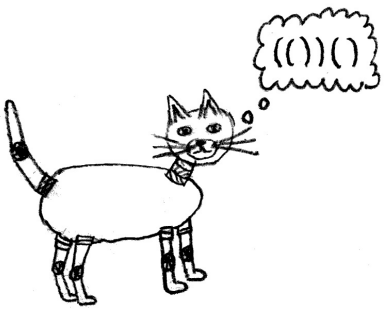
E1. Cats on the Upgrade (easy version)

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is the easy version of the problem. The only difference between the easy and the hard versions are removal queries, they are present only in the hard version.

"Interplanetary Software, Inc." together with "Robots of Cydonia, Ltd." has developed and released robot cats. These electronic pets can meow, catch mice and entertain the owner in various ways.

The developers from "Interplanetary Software, Inc." have recently decided to release a software update for these robots. After the update, the cats must solve the problems about bracket sequences. One of the problems is described below.



First, we need to learn a bit of bracket sequence theory. Consider the strings that contain characters "(", ")" and ".". Call a string *regular bracket sequence (RBS)*, if it can be transformed to an empty string by one or more operations of removing either single "." characters, or a continuous substring "()". For instance, the string "(()(.))" is an RBS, as it can be transformed to an empty string with the following sequence of removals:

"(()(.))" $\rightarrow$ "(()())" $\rightarrow$ "(())" $\rightarrow$ "()" $\rightarrow$ "".

We got an empty string, so the initial string was an RBS. At the same time, the string ")(" is not an RBS, as it is not possible to apply such removal operations to it.

An RBS is *simple* if this RBS is not empty, doesn't start with ".", and doesn't end with ".".

Denote the *substring* of the string s as its sequential subsegment. In particular, $s[l \ldots r] = s_l s_{l+1} \ldots s_r$, where s_i is the i-th character of the string s.

Now, move on to the problem statement itself. You are given a string s, initially consisting of characters "(" and ")". You need to answer the queries of the following kind.

Given two indices, l and r ($1 \leq l < r \leq n$), and it's **guaranteed** that the substring $s[l \ldots r]$ is a **simple RBS**. You need to find the number of substrings in $s[l \ldots r]$ such that they are simple RBS. In other words, find the number of index pairs i, j such that $l \leq i < j \leq r$ and $s[i \ldots j]$ is a simple RBS.

You are an employee in "Interplanetary Software, Inc." and you were given the task to teach the cats to solve the problem above, after the update.

Note that the "." character cannot appear in the string in this version of the problem. It is only needed for the hard version.

Input

The first line contains two integers n and q ($2 \leq n \leq 3 \cdot 10^5$, $1 \leq q \leq 3 \cdot 10^5$), the length of the string, and the number of queries.

The second line contains the string s, consisting of n characters "(" and ")".

Each of the following q lines contains three integers t, l and r ($t = 2$, $1 \leq l < r \leq n$), the queries you need to answer. It is guaranteed that all the queries are valid and correspond to the problem statements. **Note that** t is unused and always equal to two

in this problem. It is required for the hard version of the problem.

Output

For each query, print a single integer in a separate line, the number of substrings that are simple RBS. The answers must be printed in the same order as the queries are specified in the input.

Example

input
9 4)(()())() 2 3 6 2 2 7 2 8 9 2 2 9
output
3 4 1 6

Note

Consider the example test case.

The answer to the first query is 3, as there are three suitable substrings: $s[3 \dots 6]$, $s[3 \dots 4]$ and $s[5 \dots 6]$.

The answer to the second query is 4. The substrings are $s[3 \dots 6]$, $s[3 \dots 4]$, $s[5 \dots 6]$ and $s[2 \dots 7]$.

The answer to the third query is 1. The substring is $s[8 \dots 9]$.

The answer to the fourth query is 6. The substrings are $s[3 \dots 6]$, $s[3 \dots 4]$, $s[5 \dots 6]$, $s[2 \dots 7]$, $s[8 \dots 9]$ and $s[2 \dots 9]$.

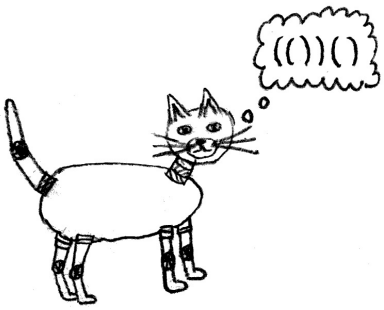
E2. Cats on the Upgrade (hard version)

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is the hard version of the problem. The only difference between the easy and the hard versions are removal queries, they are present only in the hard version.

"Interplanetary Software, Inc." together with "Robots of Cydonia, Ltd." has developed and released robot cats. These electronic pets can meow, catch mice and entertain the owner in various ways.

The developers from "Interplanetary Software, Inc." have recently decided to release a software update for these robots. After the update, the cats must solve the problems about bracket sequences. One of the problems is described below.



First, we need to learn a bit of bracket sequence theory. Consider the strings that contain characters "(", ")" and ".". Call a string *regular bracket sequence (RBS)*, if it can be transformed to an empty string by one or more operations of removing either single "." characters, or a continuous substring "()". For instance, the string "(( ) (.))" is an RBS, as it can be transformed to an empty string with the following sequence of removals:

$$\text{"(( ) (.))"} \to \text{"(( ) ( ))"} \to \text{"(( ))"} \to \text{"( )"} \to \text{""}.$$

We got an empty string, so the initial string was an RBS. At the same time, the string ") (" is not an RBS, as it is not possible to apply such removal operations to it.

An RBS is *simple* if this RBS is not empty, doesn't start with ".", and doesn't end with ".".

Denote the *substring* of the string s as its sequential subsegment. In particular, $s[l \dots r] = s_l s_{l+1} \dots s_r$, where s_i is the i-th character of the string s.

Now, move on to the problem statement itself. You are given a string s, initially consisting of characters "(" and ")". You need to answer the following queries:

1. Given two indices, l and r ($1 \le l < r \le n$). It's **guaranteed** that the l-th character is equal to "(", the r-th character is equal to ")", and the characters between them are equal to ".". Then the l-th and the r-th characters must be set to ".".

2. Given two indices, l and r ($1 \le l < r \le n$), and it's **guaranteed** that the substring $s[l \dots r]$ is a **simple RBS**. You need to find the number of substrings in $s[l \dots r]$ such that they are simple RBS. In other words, find the number of index pairs i, j such that $l \le i < j \le r$ and $s[i \dots j]$ is a simple RBS.

You are an employee in "Interplanetary Software, Inc." and you were given the task to teach the cats to solve the problem above, after the update.

Input

The first line contains two integers n and q ($2 \le n \le 3 \cdot 10^5$, $1 \le q \le 3 \cdot 10^5$), the length of the string, and the number of queries.

The second line contains the string s, consisting of n characters "(" and ")".

Each of the following q lines contains three integers t, l and r ($t \in \{1, 2\}$, $1 \le l < r \le n$), the queries you need to answer. It is guaranteed that all the queries are valid and correspond to the problem statements.

Output

For each query, print a single integer in a separate line, the number of substrings that are simple RBS. The answers must be printed in the same order as the queries are specified in the input.

Example

input
9 8)(()())() 2 3 6 2 2 7 1 3 4 2 2 7 2 2 9 1 5 6 1 2 7 2 8 9
output
3 4 2 4 1

Note

Consider the example test case.

The answer to the first query is 3, as there are three suitable substrings: $s[3 \dots 6]$, $s[3 \dots 4]$ and $s[5 \dots 6]$.

The answer to the second query is 4. The substrings are $s[3 \dots 6]$, $s[3 \dots 4]$, $s[5 \dots 6]$ and $s[2 \dots 7]$.

After the third query, the string becomes ") (. . ( )) ( ) ( )".

The answer to the fourth query is 2. The substrings are $s[5 \dots 6]$ and $s[2 \dots 7]$. Note that $s[3 \dots 6]$ is not a simple RBS anymore, as it starts with ".".

The answer to the fifth query is 4. The substrings are $s[5 \dots 6]$, $s[2 \dots 7]$, $s[8 \dots 9]$ and $s[2 \dots 9]$.

After the sixth query, the string becomes ") (. . . .) ( )".

After the seventh query, the string becomes ") . . . . . ( )".

The answer to the eighth query is 1. The substring is $s[8 \dots 9]$.