Problem Barcelona. Median mountain range

Input file:	`input.txt` or standard input
Output file:	`output.txt` or standard output
Time limit:	2 seconds
Memory limit:	512 megabytes

Berland — is a huge country with diverse geography. One of the most famous natural attractions of Berland is the "Median mountain range". This mountain range is n mountain peaks, located on one straight line and numbered in order of 1 to n. The height of the i-th mountain top is a_i.

"Median mountain range' is famous for the so called *alignment of mountain peaks* happening to it every day. At the moment of alignment simultaneously for each mountain from 2 to $n-1$ its height becomes equal to the median height among it and two neighboring mountains. Formally, if before the alignment the heights were equal b_i, then after the alignment new heights a_i are as follows: $a_1 = b_1$, $a_n = b_n$ and for all i from 2 to $n-1$ $a_i = \texttt{median}(b_{i-1}, b_i, b_{i+1})$. The median of three integers is the second largest number among them. For example, $\texttt{median}(5, 1, 2) = 2$, and $\texttt{median}(4, 2, 4) = 4$.

Recently, Berland scientists have proved that whatever are the current heights of the mountains, the alignment process will stabilize sooner or later, i.e. at some point the altitude of the mountains won't changing after the alignment any more. The government of Berland wants to understand how soon it will happen, i.e. to find the value of c — how many alignments will occur, which will change the height of at least one mountain. Help scientists solve this important problem!

Note that in some test groups, in addition to the c value, you will need to determine the mountain heights after the c alignments, i.e. the heights of the mountains as they will stay forever.

Input

The first line contains integers n and t ($1 \le n \le 500\,000$, $0 \le t \le 1$) — the number of mountains and the parameter describing whether it's required to determine the final heights of the mountains.

The second line contains integers $a_1, a_2, a_3, \ldots, a_n$ ($1 \le a_i \le 10^9$) — current heights of the mountains.

Output

In the first line print c — the number of alignments, which change the height of at least one mountain.

In case $t = 1$, the second line should contain n integers — the final heights of the mountains after c alignments.

Examples

input	output

`5 1`
`1 2 1 2 1`
	`2`
	`1 1 1 1 1`

`6 1`
`1 3 2 5 4 6`
	`1`
	`1 2 3 4 5 6`

`6 0`
`1 1 2 2 1 1`
	`0`

Note

In the first example, the heights of the mountains at index 1 and 5 never change. Since the median of 1, 2, 1 is 1, the second and the fourth mountains will have height 1 after the first alignment, and since the median of 2, 1, 2 is 2, the third mountain will have height 2 after the first alignment. This way, after one alignment the heights are 1, 1, 2, 1, 1. After the second alignment the heights change into 1, 1, 1, 1, 1 and never change from now on, so there are only 2 alignments changing the mountain heights.

In the third examples the alignment doesn't change any mountain height, so the number of alignments changing any height is 0. Since $t = 0$, you shouldn't print the resulting heights of the mountains.

Scoring

Tests for this problem are divided into six groups. For each of the groups you earn points only if your solution passes **all** tests in this group and all tests of the **required** groups. **Offline evaluation** means that your submission will be evaluated on the tests of the group only after the end of the contest.

Group	Points	Additional constraints			Req. groups	Comment
		n	a_i	t		
0	0	–	–	–	–	Example tests.
1	19	$n \le 1000$	–	–	0	It's guaranteed, that $c \le 10\,000$.
2	24	–	$a_i \le 2$	–	–	
3	14	–	$a_i \le 100$	$t = 0$	–	
4	14	–	$a_i \le 100$	–	0, 2, 3	
5	14	–	–	$t = 0$	3	
6	15	–	–	–	0, 1, 2, 3, 4, 5	**Offline evaluation.**

Problem Wimbledon. Unusual competitions

Input file:	`input.txt` or standard input
Output file:	`output.txt` or standard output
Time limit:	1 second
Memory limit:	512 megabytes

The teacher gave Dmitry's class a very strange task — she asked every student to come up with a sequence of arbitrary length, consisting only of opening and closing brackets. After that all the students took turns naming the sequences they had invented. When the Dima's turn come, he suddenly realized that all his classmates got the right bracketed sequence, and whether he got the right bracketed sequence, he did not know.

Dima suspects now that he simply missed the word "right" in the task statement, so now he wants to save the situation by modifying his sequence slightly. More precisely, he can arbitrary amount of times (possibly zero) perform the *reorder* operation. The reorder operation consists of choosing an arbitrary subsegment of the sequence and then reordering all the characters in it in an arbitrary way. Such operation takes l nanoseconds, where l is the length of the subsegment being reordered. It's easy to see that reorder operation doesn't affect the number of opening and closing brackets doesn't change.

Since Dima will soon have to answer, he wants to make his sequence right as fast as possible. Help him to do this, or determine that it's impossible.

Input

The first line contains a single integer n ($1 \le n \le 10^6$) — the length of Dima's sequence.

The second line contains string of length n, consisting of characters "`(`" and "`)`" only.

Output

Print a single integer — the minimum number of nanoseconds to make the sequence right or "`-1`" if it is impossible to do so.

Examples

input	output

`8`
`))((())(`
	`6`

`3`
`(()`
	`-1`

Note

A bracketed sequence is called right if by inserting "+" and "1" you can get a well-formed mathematical expression from it. For example, sequences "`(())()`", "`()`" and "`(()(()))`" are right, while "`)(`", "`(()`" and "`(()))(`" are not.

In the first example we can firstly reorder the segment from first to the fourth character, replacing it with "`()()`", the whole sequence will be "`()()())(`". And then reorder the segment from the seventh to eighth character, replacing it with "`()`". In the end the sequence will be "`()()()()`", while the total time spent is $4 + 2 = 6$ nanoseconds.

Scoring

Tests for this problem are divided into two groups. For each of the groups you earn points only if your solution passes **all** tests in this group and all tests of the **previous** groups.

Group	Points	Additional constraints	Comments
		n	
0	0	–	Example tests.
1	50	$n \le 1000$	
2	50	–	

Problem Zermatt. Assigning Fares

Input file:	`input.txt` or standard input
Output file:	`output.txt` or standard output
Time limit:	6 seconds
Memory limit:	512 megabytes

Mayor of city M. decided to launch several new metro lines during 2020. Since the city has a very limited budget, it was decided not to dig new tunnels but to use the existing underground network.

The tunnel system of the city M. consists of n metro stations. The stations are connected with $n-1$ bidirectional tunnels. Between every two stations v and u there is exactly one simple path. Each metro line the mayor wants to create is a simple path between stations a_i and b_i. Metro lines can intersects freely, that is, they can share common stations or even common tunnels. However, it's not yet decided which of two directions each line will go. More precisely, between the stations a_i and b_i the trains will go either from a_i to b_i, or from b_i to a_i, but not simultaneously.

The city M uses complicated faring rules. Each station is assigned with a positive integer c_i — the fare zone of the station. The cost of travel from v to u is defined as $c_u - c_v$ roubles. Of course, such travel only allowed in case there is a metro line, the trains on which go from v to u. Mayor doesn't want to have any travels with a negative cost, so it was decided to assign directions of metro lines and station fare zones in such a way, that fare zones are strictly increasing during any travel on any metro line.

Mayor wants firstly assign each station a fare zone and then choose a lines direction, such that all fare zones are increasing along any line. In connection with the approaching celebration of the day of the city, the mayor wants to assign fare zones so that the maximum c_i will be as low as possible. Please help mayor to design a new assignment or determine if it's impossible to do. Please note that you only need to assign the fare zones optimally, you don't need to print lines' directions. This way, you solution will be considered correct if there will be a way to assign directions of every metro line, so that the fare zones will be strictly increasing along any movement of the trains.

Please note, that in some groups it's not required to minimize the answer, you only need to determine whether it's possible to assign fare zones in a valid way.

Input

The first line contains an integers n, m, t ($2 \le n, \le 500\,000,\ 1 \le m \le 500\,000,\ 0 \le t \le 1$) — the number of stations in the city, the number of metro lines, and the parameter t. In case $t = 0$ you don't have to minimize the answer. In case $t = 1$ you should find an answer with the largest fare zone being the smallest possible.

Each of the following $n-1$ lines describes another metro tunnel. Each tunnel is described with integers v_i, u_i ($1 \le v_i,\ u_i \le n$, $v_i \ne u_i$). It's guaranteed, that there is exactly one simple path between any two stations.

Each of the following m lines describes another metro line. Each line is described with integers a_i, b_i ($1 \le a_i,\ b_i \le n$, $a_i \ne b_i$).

Output

In the first line print integer k — the maximum fare zone used. In case parameter t is 0, you don't have to minimize k. In case $t = 1$, the k should be the smallest possible.

In the next line print integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le k$) — stations' fare zones.

In case there are several possible answers, print any of them. In case it's impossible to assign fare zones, print "`-1`".

Examples

input	output

`3 1 1`
`1 2`
`2 3`
`1 3`
	`3`
	`1 2 3`

`4 3 0`
`1 2`
`1 3`
`1 4`
`2 3`
`2 4`
`3 4`
	`-1`

Note

In the first example, line $1 \to 3$ goes through the stations 1, 2, 3 in this order. In this order the fare zones of the stations are increasing. Since this line has 3 stations, at least three fare zones are needed. So the answer 1, 2, 3 is optimal.

In the second example, it's impossible to assign fare zones to be consistent with a metro lines.

Scoring

Tests for this problem are divided into ten groups. For each of the groups you earn points only if your solution passes all tests in this group and all tests in required groups. **Offline evaluation** means that your submission will be evaluated on the tests of the group only after the end of the contest.

In the groups 4 and 5 there is no tunnel, which belongs to two different lines simultaneously.

The length of the metro line is the number of tunnels it contains.

Group	Points	Additional constraints		Req. groups	Comment
		n, m	t		
0	0	–	–	–	Example tests.
1	6	$n,\ m \le 8$	–	0	
2	10	$n,\ m \le 15$	–	0, 1	
3	15	$n,\ m \le 100$	–	0, 1, 2	
4	11	$n,\ m \le 100\,000$	–	–	No common tunnels.
5	10	–	–	4	No common tunnels.
6	8	–	–	–	All tunnels have a common endpoint.
7	10	$n,\ m \le 100\,000$	$t = 0$	–	Total lines length is at most 100 000.
8	6	–	$t = 0$	7	
9	14	$n,\ m \le 100\,000$	–	0, 1, 2, 3, 4, 7	
10	10	–	–	0 – 9	**Offline evaluation.**

Problem Dakar. Double Palindrome

Input file:	`input.txt` or standard input
Output file:	`output.txt` or standard output
Time limit:	2 seconds
Memory limit:	512 megabytes

Vanya works at the factory producing palindromes. The factory has a workpiece — a string s line of length n, consisting of lowercase English letters, from which Vanya can cut out any substring for sale. We remind you that *palindrome* — is a string that reads in the same way from left to right and from right to left.

A lot of people are fed up with a usual palindromes, so Vanya decided to produce double palindromes instead. *Double palindrome* is a string formed by a concatenation of two palindromes of **equal** length. For example, the strings "`aabb`", "`aaaa`" are double palindromes, while strings "`abba`" and "`aaaabb`" are not.

Vanya wonders how many ways are there to cut out double palindrome from s. In other words, how many there are pairs (l, r), such that substring $s_l s_{l+1} \dots s_r$ is a double palindrome. Please help Vanya to find an answer to this question.

Input

The first line contains an integer n ($1 \le n \le 500\,000$) — the length of the string s. The second contains a string s, consisting of lowercase English letters.

Output

Print a single integer — the number of double palindrome substrings.

Examples

input	output
6 abacac	6
5 aaaaa	6

Note

In the first example, there are 5 double palindromes of length 2 ("`ab`", "`ba`", "`ac`", "`ca`" and "`ac`"), and the whole string is a double palindrome as well ("`abacac`").

Scoring

Tests for this problem are divided into three groups. For each of the groups you earn points only if your solution passes **all** tests in this group and all tests of the **previous** groups.

Group	Points	Additional constraints	Comment
		n	
0	0	–	Example tests.
1	19	$n \le 500$	
2	33	$n \le 5000$	
3	48	–	