Information

Memory limit

The limit is 512 MiB for each problem.

Source code limit

The size of each solution source code can't exceed 256 KiB.

Submissions limit

You can submit at most 50 solutions for each problem.

You can submit a solution to each task at most once per 30 seconds. This restriction does not apply in the last 15 minutes of the contest round.

Scoring

Each problem consists of several subtasks. The subtask score is awarded if all tests in the subtask are passed.

The number of points scored for the problem is the total number of points scored on each of its subtasks. The score for the subtask is the maximum number of points earned for this subtask among all the solutions submitted.

Feedback

To get feedback for your solution, go to "Runs" tab in PCMS2 Web Client and use "View Feedback" link. In each problem of the contest you will see the score for each subtask, or the verdict for the first failed test.

Scoreboard

The contestants' scoreboard is available during the contest. Use "Monitor" link in PCMS2 Web Client to access the scoreboard. The standings provided in PCMS2 Web Client are not final.

Problem A. Universal Paperclips

Time limit:	2 seconds

Universal Paperclips is an incremental game whose goal is to produce paperclips. At your disposal, you have a device that makes paperclips at the click of a button.

For this problem, we'll consider a simplified version of the game with two types of actions:

1. Click the button. Each click adds a certain amount to the current number of paperclips. This action is denoted by letter "`C`" ("click").

2. Upgrade the paperclip-making device. After this action, every subsequence button click will produce one more paperclip. This action is denoted by the letter "`U`" ("upgrade").

Initially, the player has zero paperclips, and each button click produces one paperclip. An upgrade costs x paperclips. **In all test cases, it's guaranteed that before every upgrade action, the player has at least x paperclips.**

Benny is coding a bot that plays this game. He programmed a fixed sequence of n actions S, that bot executes cyclically. Every second, the bot makes another action: on the 1st second, the bot executes action S_1, on the 2nd — S_2, …, on the n-th second it executes S_n, on the $n+1$-th second — S_1 again, and then the cycle repeats.

How many paperclips will Benny have after his bot runs for t seconds?

Input

The first line contains three integers n, t and x ($1 \le n \le 10^5$; $1 \le t \le 10^9$; $0 \le x < n$). The second line contains the string S of length n, consisting of characters "`C`" and "`U`".

Output

Output a single integer — the number of paperclips after t seconds.

Scoring

Subtask	Points	Constraints
1	33	$t \le 10^5$
2	67	No additional constraints

Examples

standard input	standard output

`10 15 2`
`CCCCCUUCCU`
	`25`

`3 10 0`
`UUU`
	`0`

`9 150 1`
`CCUCCCCUC`
	`2023`

Note

Let's go through the first example.

The first five actions produce one paperclip each. Then, the two subsequent upgrades cost two paperclips, so you have only one paperclip left. But, the next two clicks produce three paperclips. The next upgrade costs two paperclips, again. Each one of the last five clicks adds four paperclips.

The final number of paperclips is $1+1+1+1+1-2-2+3+3-2+4+4+4+4+4 = 25$.

Problem B. Hanoi Chips

Time limit: 2 seconds

In the city of Hanoi, in addition to the famous tower of Brahma, there is also the number line of Brahma. There are three identical chips placed in integer coordinates somewhere on this line. You can move these chips according to the following rule: take any chip (one at a time) and place it in the new position, which must be symmetrical relative to one of the two other chips. For example, if chips' coordinates are 1, 4, and 6, you can move the chip from 4 to either -2 (symmetrically relative to the first chip) or to 8 (symmetrically relative to the third chip). Multiple chips can occupy the same spot.

According to one of the legends, treasures will rain from the sky as soon as chips occupy certain positions on that line. Of course, nobody knows the exact coordinates where the chips should be, but you have a guess.

Given the starting and final arrangement of the chips, determine if the final arrangement is reachable, and, if possible, find a sequence of moves. The number of moves might not be optimal, but should not exceed 10^5. You cannot move a chip into a coordinate greater than 10^9 by absolute value.

Input

The first line of the input contains three integers x_1, x_2, x_3 ($-10^4 \le x_i \le 10^4$) — the starting arrangement of the three chips.

The second line of the input contains three integers y_1, y_2, y_3 ($-10^4 \le y_i \le 10^4$) — the final arrangement.

Output

If it's impossible to achieve the final arrangement, output a single integer "-1" (without quotes).

Otherwise, output a single integer k ($0 \le k \le 10^5$) — the number of moves. Then, print all moves (each one in a separate line) in the following format: the first number is the coordinate of the chip being moved, and the second is its new position. All coordinates should not exceed 10^9 by absolute value.

Scoring

Subtask	Points	Constraints
1	17	Both in the starting and final arrangement, three chips occupy three consecutive spots (for example, $3, 4, 5$ or $-7, -9, -8$)
2	20	$0 \le x_i, y_i \le 50$
3	25	The final arrangement is reachable by at most ≤ 8 moves
4	38	No additional constraints

Examples

standard input	standard output

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

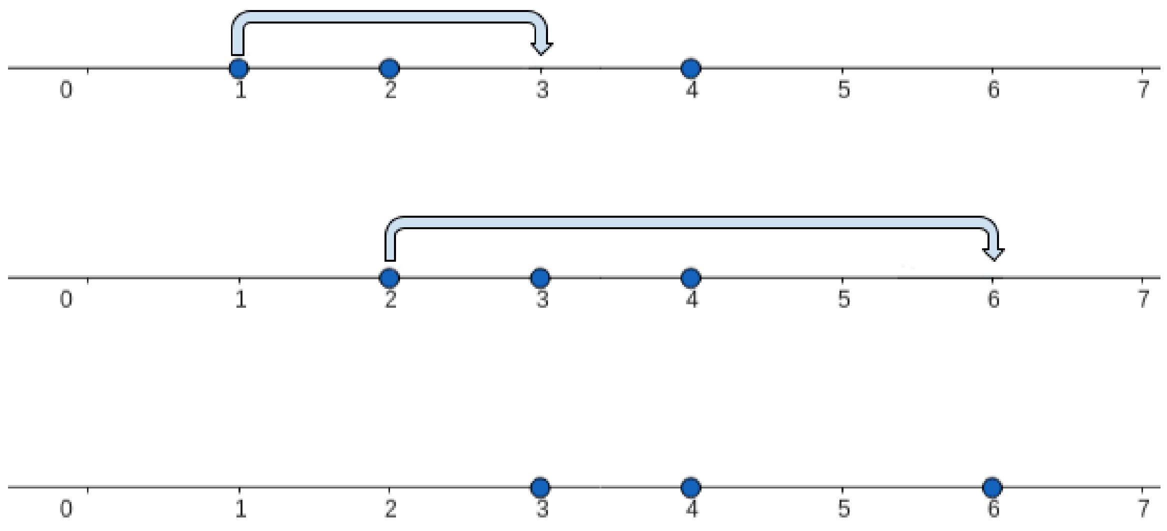
`1 2 2`
`1 2 3`
	`-1`

`1 4 5`
`-3 -2 -1`
	`4`
	`4 -2`
	`5 -3`
	`-3 -1`
	`1 -3`

Note

Please note that since the chips are identical, the chips do not have to finish at the corresponding coordinates in the end. For example, x_1 can go to y_2, x_2 to y_3 and x_3 to y_1.

In the first example, you need to move chips from the coordinates $(1, 2, 4)$ to the coordinates $(3, 4, 6)$. To achieve this, you can make two moves: move the chip from 1 to 3 and then another chip from 2 to 6.



Problem C. Sorting Subarrays

Time limit: 2 seconds

You are given an array of integers. You perform the following action exactly once: choose a non-empty subsegment of the array and order its elements in non-decreasing order.

How many different arrays can you obtain? A subsegment of an array is a collection of several consecutive elements.

Input

The first line contains a single integer n — the size of the array ($1 \le n \le 200\,000$).

The second line contains n integers a_1, a_2, $\ldots a_n$ — the contents of the array ($1 \le a_i \le 10^9$).

Output

Output a single integer — the number of different arrays you can obtain by sorting a subsegment once.

Scoring

Subtask	Points	Constraints
1	13	$n \le 50$
2	26	$n \le 3\,000$
3	22	$1 \le a_i \le 2$
4	39	No additional constraints

Examples

standard input	standard output
7 3 1 4 1 5 9 2	8
4 1 2 3 4	1

Note

Listed below are all eight achievable arrays in the first example. Square brackets indicate the sorted subsegment.

1. $[3], 1, 4, 1, 5, 9, 2 \to 3, 1, 4, 1, 5, 9, 2$
2. $[3, 1], 4, 1, 5, 9, 2 \to 1, 3, 4, 1, 5, 9, 2$
3. $[3, 1, 4, 1], 5, 9, 2 \to 1, 1, 3, 4, 5, 9, 2$
4. $[3, 1, 4, 1, 5, 9, 2] \to 1, 1, 2, 3, 4, 5, 9$
5. $3, [1, 4, 1], 5, 9, 2 \to 3, 1, 1, 4, 5, 9, 2$
6. $3, [1, 4, 1, 5, 9, 2] \to 3, 1, 1, 2, 4, 5, 9$
7. $3, 1, 4, [1, 5, 9, 2] \to 3, 1, 4, 1, 2, 5, 9$
8. $3, 1, 4, 1, 5, [9, 2] \to 3, 1, 4, 1, 5, 2, 9$

Problem D. RestORe

Time limit: 2 seconds

Consider a sequence of non-negative integers from L to R inclusive. Split this sequence into n non-empty segments and calculate the bitwise "OR" of all numbers in each segment. Denote the result for the i-th segment as f_i.

You need to solve the inverse problem: given $f_1, f_2, \ldots, f_n$, find the number of ways first to select L and R, and then split all numbers from L to R into n non-empty segments, such that the results of calculating the bitwise "OR" in each segment are equal to $f_1, f_2, \ldots, f_n$ respectively.

The bitwise "OR" of two non-negative integers is calculated as follows: Write down both arguments in binary, then the i-th bit of the result equals 1 iff at least one argument has 1 at that place. For example, $(17 \mid 13) = (10001_2 \mid 01101_2) = 11101_2 = 29$.

Input

The first line contains integer n ($1 \le n \le 200\,000$). The second line contains n integers f_i ($0 \le f_i < 2^{60}$).

Output

Output a single integer — the number of ways to choose a sequence of consecutive integers and then split it into n segments. Because the answer can be huge, print it modulo $1\,000\,000\,007$.

Scoring

Subtask	Points	Constraints
1	16	$n = 1$
2	12	$n \le 16$, $0 \le f_i < 16$
3	13	$n \le 256$, $0 \le f_i < 256$
4	18	$n \le 1024$, $0 \le f_i < 1024$
5	24	$n \le 30\,000$, $0 \le f_i < 2^{30}$
6	17	No additional constraints

Examples

standard input	standard output
3 0 7 7	4
1 11	6
2 4 6	0

Note

In the first example, there are four ways to split a sequence of consecutive integers into three parts such that their bitwise "OR"s are 0, 7 and 7:

- $[0], [1, 2, 3, 4, 5, 6], [7]$: $0 = 0$; $1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 = 7$; $7 = 7$
- $[0], [1, 2, 3, 4, 5], [6, 7]$: $0 = 0$; $1 \mid 2 \mid 3 \mid 4 \mid 5 = 7$; $6 \mid 7 = 7$
- $[0], [1, 2, 3, 4], [5, 6, 7]$: $0 = 0$; $1 \mid 2 \mid 3 \mid 4 = 7$; $5 \mid 6 \mid 7 = 7$
- $[0], [1, 2, 3, 4], [5, 6]$: $0 = 0$; $1 \mid 2 \mid 3 \mid 4 = 7$; $5 \mid 6 = 7$

Problem E. Non-adjacent Swaps

Time limit: 2 seconds

You are given a permutation. In one step, you can swap any two adjacent elements if their values differ by more than one.

Consider all permutations that can be obtained using these operations, starting from a given permutation. Define a *distance* between two permutations as the minimum number of these operations required to transform the first permutation into the second one.

Find the number of permutations that can be obtained and the sum of distances to all those permutations from the starting one.

Input

The first line contains the integer n ($1 \le n \le 100$) — the permutation size. The second line contains n distinct integers a_i ($1 \le a_i \le n$) — the starting permutation.

Output

Output two integers, the number of obtainable permutations, and the sum of distances to those permutations. Calculate both numbers modulo 1 000 000 007.

Scoring

Subtask	Points	Constraints
1	15	$n \le 8$
2	20	$n \le 15$
3	30	$n \le 30$
4	20	$n \le 50$
5	15	No additional constraints

Examples

standard input	standard output
4 3 1 4 2	5 5
5 1 2 3 4 5	1 0
6 5 3 1 2 4 6	61 218

Note

In the first example, you can obtain the following permutations:

- $[3, 1, 4, 2]$, in 0 steps;
- $[3, 1, 2, 4]$, in 1 step;
- $[1, 3, 4, 2]$, in 1 step;
- $[3, 4, 1, 2]$, in 1 step;
- $[1, 3, 2, 4]$, in 2 steps (swap two pairs of elements with values $(3, 1)$ and $(4, 2)$).

There are five permutations in total; the sum of distances also equals five.

In the second example, you can't perform any swap, so only one permutation is reachable with a distance of 0.