Problem name: Andres Iniesta

Language: English

Time limit: **1 s**
Memory limit: **128 MB**

Task description

As you know Spain won't defend World Cup champion title, but they already started to train for Euro 2016. One of their best players is Andres Iniesta. As you know, midfielders should have perfect pass and they have to move on the field in such a way that their field of view is maximized.

Spain's coach, Vicente del Bosque, invented a new training drill for midfielders. The training field is represented by a NxM matrix with obstacles positioned on some cells. Player can stand on any cell in the matrix that doesn't contain obstacle and player's field of view is calculated by the number of cells he can see. If the player stands on a cell then he can see another cell if those cells are located at the **same row or same column and there is no obstacle between cells (including those cells)**. Iniesta can see a cell on which he is standing.

As you know, Iniesta has a perfect pass, so he can destroy any obstacle by shooting a ball at it from any position.

Vicente gives K balls to Iniesta, so he can destroy up to K obstacles. After destroying obstacles his job is to find a cell on which his field of view is maximized.

Iniesta lost the focus after Spain lost versus Netherland by $5 - 1$ at the start of World Cup, so he asks you for help. Your job is calculate what is the maximum field of view if Iniesta can destroy up to K obstacles and after destroying he can stand on any cell that doesn't contain obstacle.

Input

First line of input contains numbers N, M and K, that represent number of rows, number of columns in the matrix and the number of balls that are given to Iniesta, respectively. Each of the following N lines contains M characters representing a training field in a form of a matrix. Cell located at row r and column c equals $'*'$ if there is an obstacle at row r and cell c on the field. Cell that doesn't contain obstacle is represented in the matrix by a character $'.'$.

Output

In the first line of output you should print what is the maximum field of view after destroying up to K obstacles and standing on a cell that doesn't contain obstacle. Iniesta can stand on a cell if it was empty at beginning or if an obstacle is destroyed on that cell.

Example

Input:	Output:
7 4 5 ***. **.* ..*. **.* **** *.** ****	10

Example explanation

After destroying obstacles at positions $\{(1,3),(3,3),(5,3),(6,3),(7,3)\}$ in matrix (rows and columns indexed from 1), Iniesta should stand at the cell $(3,3)$ in the matrix, and he can see 10 cells as it is shown in the Figure 1.

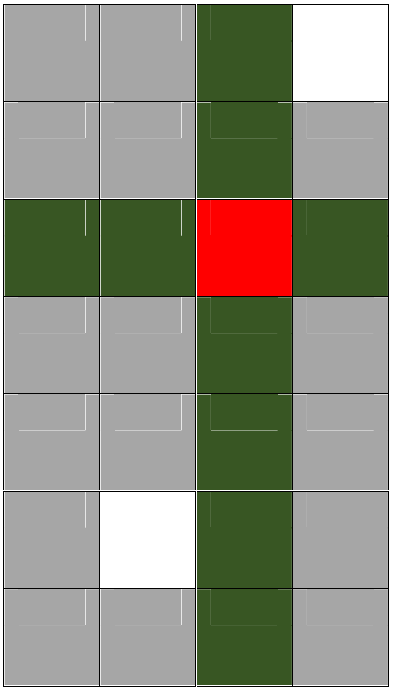


Figure 1

Red cell represents where Iniesta should stand after destroying obstacles. Green cells represent cells that Iniesta can see. Grey cells are cells with obstacles. White cells are empty cells that Iniesta can't see from his position.

Constraints

- $1 \leq N, M \leq 200$
- $0 \leq K \leq 50$
- In first 20% of test cases it will hold $N, M \leq 10, K \leq 20$
- In next 10% of test cases it will hold $K = 0$
- In next 20% of test cases it will hold $1 \leq K \leq 2$

Problem name: Anthem

Language: English

Time limit: **0.1 s**
Memory limit: **64 MB**

Task description

At the World cup, before every game, the whole team, including substitutes, **stands in line** while singing their national anthem. There is an **even number of players** in a team. The team from Bitland was very nervous before their first game so it looks like they mixed up the jerseys.

They need to find out the numbers on their backs before the anthem finishes, so they agreed that every player should say the **sum of numbers which he sees**. Every player can see the numbers on the backs of his **immediate teammates which are placed left and right from him**. The first (last) player in a line can only see the number of the teammate on his left (right) side. Help them determine numbers on their backs before the game, if you know all of the sums that they said.

Input

The input contains two lines. The first line contains one **even positive integer** N, which represents the number of players in the team. The second line contains N positive integers, separated by one space, representing the **numbers that players said**.

Output

The first and only line of the output contains N integer number, separated by one space, which are the numbers on the jerseys. The numbers should be presented in the same order as in input. If there are multiple solutions, output any one of them.

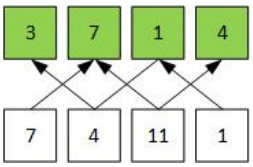
Example

Input:	Output:

4 7 4 11 1	3 7 1 4

Example description

The first player can only see the number of the second player, and from output we have $7 = 7$. Second player can see numbers for first and third players, and from output have $4 = 3 + 1$. Third player can see numbers on second and forth player, and from output we have $11 = 7 + 4$. Finally, fourth player can only see number of third player, and from output we have $1 = 1$.



Constraints

- $2 \leq N \leq 100.000$
- Numbers on jerseys are from the segment $[1, 10^6]$.
- There can be jerseys with the same numbers on their backs.
- It is guaranteed that a solution will always exist.

Problem name: Mascot Song

Language: English

Time limit: **0.5 s**
Memory limit: **32 MB**

Task description

As we all know, every FIFA World Cup has a mascot and every mascot has The Official Mascot Song. For the current World Cup in Brazil, the mascot is Fuleco the Armadillo and he is currently working on his song. Since he is an armadillo, his music knowledge is limited and he only knows how to play an electric guitar. Also, he writes the song in a very simple way as a sequence of n integers $A_1, A_2, \dots A_n$; the integer A_i denotes how hard will Fuleco strum the strings at the i-th second.

For some indices $1 \le i \le j \le n$, the consecutive subsequence $A_i, A_{i+1}, \dots A_j$ of a sequence $A_1, A_2, \dots A_n$ is called a **block** if the following 3 conditions hold:

1. $A_i \le A_{i-1}$ or $i = 1$,
2. $A_j \ge A_{j+1}$ or $j = n$,
3. $A_i < A_{i+1} < \dots < A_j$.

It is not hard to see that every sequence $A_1, A_2, \dots, A_n$ consists of disjoint blocks, i.e. every element A_i belongs to exactly one block. For example, the sequence $A = (\underline{3\ 4}\ \underline{4\ 5\ 7}\ \underline{3}\ \underline{2\ 3\ 10})$ consists of 4 blocks (they are underlined). Fuleco (since he is an armadillo) thinks that the quality of the song depends on the number of blocks in it. He constantly does some corrections to the song and asks you to tell him the current number of blocks; he cannot count them himself since (you can guess) he is an armadillo.

More precisely, you are given the starting sequence $A_1, A_2, \dots, A_n$ and q queries. Each query is one of the following two types:

- $1\ x\ y$ – Fuleco changes value of the x-th element of the sequence A into y, i.e. $A_x := y$.
- $2\ z$ – Fuleco cyclically shifts the whole sequence A for the z positions to the left. When some element goes over the first position, it is moved to the n-th position. For example, the query "2 3" transforms the sequence $(1, 2, 3, 4, 5)$ into $(4, 5, 1, 2, 3)$.

After each query you must return the number of blocks in a current sequence (song). Note that the queries occur in the given order and they change the starting sequence. **Sequence A is 1-indexed**.

Input

First line of input contains one integer n – the length of starting sequence (song) A. Second line contains n space separated integers A_i – the starting sequence. Third line contains one integer q – the number of queries. The next q lines represent the queries in the format described above.

Output

After each query you should write the number of blocks in a current sequence. Each number must be written in a new line and the order must be the same as the order of the queries in the input.

Example

Input:	Output:
9 3 4 4 5 7 3 2 3 10 4 1 8 9 1 7 5 2 6 1 2 3	4 3 4 5

Example explanation

After 1st query, the sequence becomes (3 4 4 5 7 3 2 **9** 10) and has 4 blocks (underlined). After 2nd query, the sequence becomes (3 4 4 5 7 3 **5** 9 10) and has 3 blocks. After 3rd query (left cyclic shift for 6 positions), we have sequence (5 9 10 3 4 4 5 7 3) with 4 blocks. Finally, after 4th query, the sequence becomes (5 **3** 10 3 4 4 5 7 3) and has 5 blocks.

Constraints

- $2 \leq n \leq 200.000$
- For each $i = \overline{1,n}$, it holds $1 \leq A_i \leq 10^9$.
- $1 \leq q \leq 200.000$
- For each query "$1\ x\ y$" it holds $1 \leq x \leq n$, $1 \leq y \leq 10^9$. For each query "$2\ z$" it holds $1 \leq z \leq n$.
- In 30% of test cases it will hold $n \leq 100$ and $q \leq 100$.
- In 30% of test cases there will be only the queries of the first type ("$1\ x\ y$").

Problem name: Tickets

Language: English

Time limit: **0.1 s**
Memory limit: **64 MB**

Task description

As we all know well, the World Cup is currently ongoing. In a few days, the Second Stage of a cup will begin. But the organizers forgot to print the tickets. In order to print the tickets, only one machine is used, and tickets are printed one by one. Using the standard method, the tickets are enumerated from 1 to N, and printed in order. However, since there is not enough time for that, the IT department proposed another solution: print the tickets in an order such that numbers on tickets represent the Gray code, because in that case changing the state of the printer from one ticket to the next one requires only one bit to be changed, and therefore the process of printing the tickets can be hugely improved.

The Gray code is a binary system where consecutive values differ in only one bit. For example, 1001 and 1011 differ in only 1 bit, but 1101 and 1011 differ in 2 bits and hence can't be consecutive in gray code. One way to construct a gray code from all numbers containing exactly n bits is the following method:

1. If $n = 1$, then the gray code is $[0, 1]$
2. Else let L_1 be the gray code of numbers containing exactly $n-1$ bits, $L_1 = [s_1, s_2, \dots, s_{2^{n-1}}]$, and let L_2 be the reversed L_1, that is $L_2 = reverse(L_1) = [s_{2^{n-1}}, s_{2^{n-1}-1}, \dots, s_2, s_1]$.
   Now let L_1' be constructed from L_1 by prefixing each element of L_1 with 0, and L_2' be constructed from L_2 by prefixing each element of L_2 with 1.
   $$L_1' = [0s_1, 0s_2, \dots, 0s_{2^{n-1}}]$$
   $$L_2' = [1s_{2^{n-1}}, 1s_{2^{n-1}-1}, \dots, 1s_2, 1s_1]$$
   Finally, the gray code of all numbers consisting of n bits, L, is constructed by appending L_2' to L_1', that is $L = L_1' + L_2' = [0s_1, 0s_2, \dots, 0s_{2^{n-1}}, 1s_{2^{n-1}}, 1s_{2^{n-1}-1}, \dots, 1s_2, 1s_1]$.

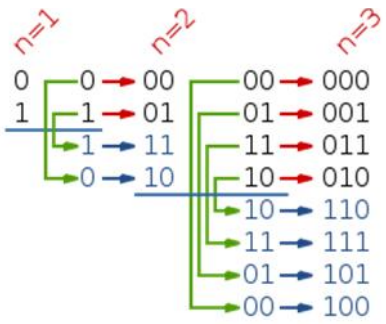


Figure 2 source: http://en.wikipedia.org/wiki/Gray_code

This indeed speeds up the process of printing the tickets by a large factor, but in order to use this method we need a safety method as well. This is needed, because the printer used is not perfect and can sometimes print with a flaw (e.g. can wrongly print some bits in input). However, the printer works correctly in 99% of cases, and therefore only a small number of tickets won't be printed correctly and those tickets will need to be printed again. For this, we need your help.

Given an number n, the number of bits in numbers, and a number K that represents the index in the gray code constructed using above method, print the K^{th} string in this gray code sequence.

Input

First and only line of the input contains two integers n and K, representing the number of bits in numbers and index in gray code sequence, respectively.

Output

First and only line of the output should contain a string of n bits, representing the K^{th} string in gray code constructed using method described in the problem statement.

Example

Input:	Output:
3 5	110

Example explanation

Gray code of numbers consisting of 3 bits constructed using the method described in problem statement is: [000, 001, 011, 010, 110, 111, 101, 100], and the 5^{th} in this sequence is 110.

Note that the sequence is indexed from 1.

Constraints

- $1 \leq n \leq 62$
- $1 \leq K \leq 2^n$
- In 30% of test cases $n \leq 20$