Codeforces Round #225 (Div. 2)

A. Coder

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

Iahub likes chess very much. He even invented a new chess piece named Coder. A Coder can move (and attack) one square horizontally or vertically. More precisely, if the Coder is located at position (x, y), he can move to (or attack) positions $(x + 1, y)$, $(x–1, y)$, $(x, y + 1)$ and $(x, y–1)$.

Iahub wants to know how many Coders can be placed on an $n \times n$ chessboard, so that no Coder attacks any other Coder.

Input
The first line contains an integer n $(1 \leq n \leq 1000)$.

Output
On the first line print an integer, the maximum number of Coders that can be placed on the chessboard.

On each of the next n lines print n characters, describing the configuration of the Coders. For an empty cell print an '`.`', and for a Coder print a '`C`'.

If there are multiple correct answers, you can print any.

Sample test(s)

input
2
output
2 C. .C

B. Multitasking

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

Iahub wants to enhance his multitasking abilities. In order to do this, he wants to sort n arrays simultaneously, each array consisting of m integers.

Iahub can choose a pair of distinct indices i and j $(1 \le i, j \le m, i \ne j)$. Then in each array the values at positions i and j are swapped only if the value at position i is strictly greater than the value at position j.

Iahub wants to find an array of pairs of distinct indices that, chosen in order, sort all of the n arrays in ascending or descending order (the particular order is given in input). The size of the array can be at most $\frac{m(m-1)}{2}$ (at most $\frac{m(m-1)}{2}$ pairs). Help Iahub, find any suitable array.

Input

The first line contains three integers n $(1 \le n \le 1000)$, m $(1 \le m \le 100)$ and k. Integer k is 0 if the arrays must be sorted in ascending order, and 1 if the arrays must be sorted in descending order. Each line i of the next n lines contains m integers separated by a space, representing the i-th array. For each element x of the array i, $1 \le x \le 10^6$ holds.

Output

On the first line of the output print an integer p, the size of the array (p can be at most $\frac{m(m-1)}{2}$). Each of the next p lines must contain two distinct integers i and j $(1 \le i, j \le m, i \ne j)$, representing the chosen indices.

If there are multiple correct answers, you can print any.

Sample test(s)

input
2 5 0 1 3 2 5 4 1 4 3 2 5

output
3 2 4 2 3 4 5

input
3 2 1 1 2 2 3 3 4

output
1 2 1

Note

Consider the first sample. After the first operation, the arrays become $[1, 3, 2, 5, 4]$ and $[1, 2, 3, 4, 5]$. After the second operation, the arrays become $[1, 2, 3, 5, 4]$ and $[1, 2, 3, 4, 5]$. After the third operation they become $[1, 2, 3, 4, 5]$ and $[1, 2, 3, 4, 5]$.

C. Milking cows

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

Iahub helps his grandfather at the farm. Today he must milk the cows. There are n cows sitting in a row, numbered from 1 to n from left to right. Each cow is either facing to the left or facing to the right. When Iahub milks a cow, all the cows that see the current cow get scared and lose one unit of the quantity of milk that they can give. A cow facing left sees all the cows with lower indices than her index, and a cow facing right sees all the cows with higher indices than her index. A cow that got scared once can get scared again (and lose one more unit of milk). A cow that has been milked once cannot get scared and lose any more milk. You can assume that a cow never loses all the milk she can give (a cow gives an infinitely amount of milk).

Iahub can decide the order in which he milks the cows. But he must milk each cow exactly once. Iahub wants to lose as little milk as possible. Print the minimum amount of milk that is lost.

Input

The first line contains an integer n $(1 \le n \le 200000)$. The second line contains n integers $a_1, a_2, ..., a_n$, where a_i is 0 if the cow number i is facing left, and 1 if it is facing right.

Output

Print a single integer, the minimum amount of lost milk.

Please, do not write the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Sample test(s)

input
4 0 0 1 0
output
1

input
5 1 0 1 0 1
output
3

Note

In the first sample Iahub milks the cows in the following order: cow 3, cow 4, cow 2, cow 1. When he milks cow 3, cow 4 loses 1 unit of milk. After that, no more milk is lost.

D. Volcanoes

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

Iahub got lost in a very big desert. The desert can be represented as a $n \times n$ square matrix, where each cell is a zone of the desert. The cell (i, j) represents the cell at row i and column j $(1 \le i, j \le n)$. Iahub can go from one cell (i, j) only down or right, that is to cells $(i + 1, j)$ or $(i, j + 1)$.

Also, there are m cells that are occupied by volcanoes, which Iahub cannot enter.

Iahub is initially at cell $(1, 1)$ and he needs to travel to cell (n, n). Knowing that Iahub needs 1 second to travel from one cell to another, find the minimum time in which he can arrive in cell (n, n).

Input

The first line contains two integers n $(1 \le n \le 10^9)$ and m $(1 \le m \le 10^5)$. Each of the next m lines contains a pair of integers, x and y $(1 \le x, y \le n)$, representing the coordinates of the volcanoes.

Consider matrix rows are numbered from 1 to n from top to bottom, and matrix columns are numbered from 1 to n from left to right. There is no volcano in cell $(1, 1)$. No two volcanoes occupy the same location.

Output

Print one integer, the minimum time in which Iahub can arrive at cell (n, n). If no solution exists (there is no path to the final cell), print -1.

Sample test(s)

input
4 2 1 3 1 4

output
6

input
7 8 1 6 2 6 3 5 3 6 4 3 5 1 5 2 5 3

output
12

input
2 2 1 2 2 1

output
-1

Note

Consider the first sample. A possible road is: $(1, 1) \rightarrow (1, 2) \rightarrow (2, 2) \rightarrow (2, 3) \rightarrow (3, 3) \rightarrow (3, 4) \rightarrow (4, 4)$.

E. Propagating tree

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

Iahub likes trees very much. Recently he discovered an interesting tree named propagating tree. The tree consists of n nodes numbered from 1 to n, each node i having an initial value a_i. The root of the tree is node 1.

This tree has a special property: when a value val is added to a value of node i, the value -val is added to values of all the children of node i. Note that when you add value -val to a child of node i, you also add -(-val) to all children of the child of node i and so on. Look an example explanation to understand better how it works.

This tree supports two types of queries:

- "$1\ x\ val$" — val is added to the value of node x;
- "$2\ x$" — print the current value of node x.

In order to help Iahub understand the tree better, you must answer m queries of the preceding type.

Input

The first line contains two integers n and m ($1 \le n, m \le 200000$). The second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le 1000$). Each of the next $n-1$ lines contains two integers v_i and u_i ($1 \le v_i, u_i \le n$), meaning that there is an edge between nodes v_i and u_i.

Each of the next m lines contains a query in the format described above. It is guaranteed that the following constraints hold for all queries: $1 \le x \le n$, $1 \le val \le 1000$.

Output

For each query of type two (print the value of node x) you must print the answer to the query on a separate line. The queries must be answered in the order given in the input.

Sample test(s)

input

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

output

```
3
3
0
```

Note

The values of the nodes are $[1, 2, 1, 1, 2]$ at the beginning.

Then value 3 is added to node 2. It propagates and value -3 is added to it's sons, node 4 and node 5. Then it cannot propagate any more. So the values of the nodes are $[1, 5, 1, -2, -1]$.

Then value 2 is added to node 1. It propagates and value -2 is added to it's sons, node 2 and node 3. From node 2 it propagates again, adding value 2 to it's sons, node 4 and node 5. Node 3 has no sons, so it cannot propagate from there. The values of the nodes are $[3, 3, -1, 0, 1]$.

You can see all the definitions about the tree at the following link: http://en.wikipedia.org/wiki/Tree_(graph_theory)