Codeforces Round #233 (Div. 1)

A. Cards

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

User ainta loves to play with cards. He has a cards containing letter "o" and b cards containing letter "x". He arranges the cards in a row, and calculates the score of the deck by the formula below.

1. At first, the score is 0.
2. For each block of contiguous "o"s with length x the score increases by x^2.
3. For each block of contiguous "x"s with length y the score decreases by y^2.

For example, if $a = 6, b = 3$ and ainta have arranged the cards in the order, that is described by string "ooxoooxxo", the score of the deck equals $2^2 - 1^2 + 3^2 - 2^2 + 1^2 = 9$. That is because the deck has 5 blocks in total: "oo", "x", "ooo", "xx", "o".

User ainta likes big numbers, so he wants to maximize the score with the given cards. Help ainta make the score as big as possible. Note, that he has to arrange all his cards.

Input

The first line contains two space-separated integers a and b ($0 \le a, b \le 10^5$; $a + b \ge 1$) — the number of "o" cards and the number of "x" cards.

Output

In the first line print a single integer v — the maximum score that ainta can obtain.

In the second line print $a + b$ characters describing the deck. If the k-th card of the deck contains "o", the k-th character must be "o". If the k-th card of the deck contains "x", the k-th character must be "x". The number of "o" characters must be equal to a, and the number of "x " characters must be equal to b. If there are many ways to maximize v, print any.

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.

Examples

input
2 3
output
-1 xoxox

input
4 0
output
16 oooo

input
0 4
output
-16 xxxx

B. Painting The Wall

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

User ainta decided to paint a wall. The wall consists of n^2 tiles, that are arranged in an $n \times n$ table. Some tiles are painted, and the others are not. As he wants to paint it beautifully, he will follow the rules below.

1. Firstly user ainta looks at the wall. If there is at least one painted cell on each row and at least one painted cell on each column, he stops coloring. Otherwise, he goes to step 2.
2. User ainta choose any tile on the wall with uniform probability.
3. If the tile he has chosen is not painted, he paints the tile. Otherwise, he ignores it.
4. Then he takes a rest for one minute even if he doesn't paint the tile. And then ainta goes to step 1.

However ainta is worried if it would take too much time to finish this work. So he wants to calculate the expected time needed to paint the wall by the method above. Help him find the expected time. You can assume that choosing and painting any tile consumes no time at all.

Input

The first line contains two integers n and m ($1 \le n \le 2 \cdot 10^3$; $0 \le m \le min(n^2, 2 \cdot 10^4)$) — the size of the wall and the number of painted cells.

Next m lines goes, each contains two integers r_i and c_i ($1 \le r_i, c_i \le n$) — the position of the painted cell. It is guaranteed that the positions are all distinct. Consider the rows of the table are numbered from 1 to n. Consider the columns of the table are numbered from 1 to n.

Output

In a single line print the expected time to paint the wall in minutes. Your answer will be considered correct if it has at most 10^{-4} absolute or relative error.

Examples

input
5 2 2 3 4 1
output
11.7669491886

input
2 2 1 1 1 2
output
2.0000000000

input
1 1 1 1
output
0.0000000000

C. Tree and Array

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

User ainta likes trees. This time he is going to make an undirected tree with n vertices numbered by integers from 1 to n. The tree is weighted, so each edge of the tree will have some integer weight.

Also he has an array t: $t[1], t[2], \ldots, t[n]$. At first all the elements of the array are initialized to 0. Then for each edge connecting vertices u and v ($u < v$) of the tree with weight c, ainta adds value c to the elements $t[u], t[u + 1], \ldots, t[v - 1], t[v]$ of array t.

Let's assume that $d(u, v)$ is the total weight of edges on the shortest path between vertex u and vertex v. User ainta calls a pair of integers x, y ($1 \le x < y \le n$) *good* if and only if $d(x, y) = t[x] + t[x + 1] + \ldots + t[y - 1] + t[y]$.

User ainta wants to make at least $\lfloor \frac{n}{2} \rfloor$ good pairs, but he couldn't make a proper tree. Help ainta to find such a tree.

Input

The first line contains a single integer n ($5 \le n \le 10^5$).

Output

Print $n - 1$ lines containing the description of the edges. The i-th line should contain three space-separated integers u_i, v_i, c_i ($1 \le u_i < v_i \le n$; $1 \le c_i \le 10^5$) — two vertices connected by the edge, and the weight of the edge.

Next print $\lfloor \frac{n}{2} \rfloor$ lines containing the good pairs. The k-th line should contain two space-separated integers x_k and y_k ($1 \le x_k < y_k \le n$). Of course, x_k, y_k must be a good pair. All pairs should be distinct — that is, for all j, k $(1 \le j < k \le \lfloor \frac{n}{2} \rfloor)$, $x_j \ne x_k$ or $y_j \ne y_k$ must be satisfied.

If there are many correct solutions, print any of them.

Examples

input
7

output
1 4 1 1 2 2 2 3 5 3 5 3 2 6 2 6 7 3 4 5 5 6 5 7

Note

$\lfloor x \rfloor$ is the largest integer not greater than x.

You can find the definition of a tree by the following link: `http://en.wikipedia.org/wiki/Tree_(graph_theory)`

You can also find the definition of the shortest path by the following link:
`http://en.wikipedia.org/wiki/Shortest_path_problem`

The tree and the array t in the sample output look like this:

D. Instant Messanger

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

User ainta decided to make a new instant messenger called "aintalk". With aintalk, each user can chat with other people. User ainta made the prototype of some functions to implement this thing.

1. `login(`u`)`: User u logins into aintalk and becomes online.
2. `logout(`u`)`: User u logouts and becomes offline.
3. `add_friend(`u`, `v`)`: User u and user v become friends. It means, u and v can talk with each other. The friendship is bidirectional.
4. `del_friend(`u`, `v`)`: Unfriend user u and user v. It means, u and v cannot talk with each other from then.
5. `count_online_friends(`u`)`: The function returns the number of friends of user u who are online at the moment.

Because the messenger is being tested by some users numbered from 1 to n, there is no register method. This means, at the beginning, some users may be online, and some users may have friends.

User ainta is going to make these functions, but before making the messenger public, he wants to know whether he is correct. Help ainta verify his code.

Input

The first line contains three space-separated integers n, m and q ($1 \le n \le 50000$; $1 \le m \le 150000$; $1 \le q \le 250000$) — the number of users, the number of pairs of friends, and the number of queries.

The second line contains an integer o ($1 \le o \le n$) — the number of online users at the beginning. The third line contains o space-separated integers $x_1, x_2, ..., x_o$ ($1 \le x_i \le n$) — the ids of the online users. It is guaranteed that these values are distinct.

Each of the next m lines contains two space-separated integers a_i and b_i ($1 \le a_i, b_i \le n$; $a_i \ne b_i$) — the ids of two users who are friends at the beginning. It is guaranteed there are no multiple friendship given in the input. Note that the friendship is bidirectional.

Next q lines describe the q queries in the format:

- "`O` u" ($1 \le u \le n$) : Call `online(`u`)`. It is guaranteed that user u was offline just before the function call.
- "`F` u" ($1 \le u \le n$) : Call `offline(`u`)`. It is guaranteed that user u was online just before the function call.
- "`A` u v" ($1 \le u, v \le n$; $u \ne v$) : Call `add_friend(`u`, `v`)`. It is guaranteed that these two users weren't friends just before the function call.
- "`D` u v" ($1 \le u, v \le n$; $u \ne v$) : Call `del_friend(`u`, `v`)`. It is guaranteed that these two users were friends just before the function call.
- "`C` u" ($1 \le u \le n$) : Call `count_online_friends(`u`)` and print the result in a single line.

Output

For each `count_online_friends(`u`)` query, print the required answer in a single line.

Examples

input
5 2 9 1 4 1 3 3 4 C 3 A 2 5 O 1 D 1 3 A 1 2 A 4 2 C 2 F 4 C 2
output
1 2 1

E. Sorting Permutations

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

We are given a permutation sequence $a_1, a_2, \ldots, a_n$ of numbers from 1 to n. Let's assume that in one second, we can choose some disjoint pairs $(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)$ and swap all a_{u_i} and a_{v_i} for every i at the same time ($1 \le u_i < v_i \le n$). The pairs are disjoint if every u_i and v_j are different from each other.

We want to sort the sequence completely in increasing order as fast as possible. Given the initial permutation, calculate the number of ways to achieve this. Two ways are different if and only if there is a time t, such that the set of pairs used for swapping at that time are different as sets (so ordering of pairs doesn't matter). If the given permutation is already sorted, it takes no time to sort, so the number of ways to sort it is 1.

To make the problem more interesting, we have k holes inside the permutation. So exactly k numbers of $a_1, a_2, \ldots, a_n$ are not yet determined. For every possibility of filling the holes, calculate the number of ways, and print the total sum of these values modulo 1000000007 $(10^9 + 7)$.

Input

The first line contains two integers n $(1 \le n \le 10^5)$ and k $(0 \le k \le 12)$. The second line contains the permutation sequence $a_1, \ldots, a_n$ $(0 \le a_i \le n)$. If a number is not yet determined, it is denoted as 0. There are exactly k zeroes. All the numbers a_i that aren't equal to zero are distinct.

Output

Print the total sum of the number of ways modulo 1000000007 $(10^9 + 7)$.

Examples

input
5 0 1 5 2 4 3
output
6

input
5 2 1 0 2 4 0
output
7