Codeforces Round #133 (Div. 2)

A. Tiling with Hexagons

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

Several ages ago Berland was a kingdom. The King of Berland adored math. That's why, when he first visited one of his many palaces, he first of all paid attention to the floor in one hall. The floor was tiled with hexagonal tiles.

The hall also turned out hexagonal in its shape. The King walked along the perimeter of the hall and concluded that each of the six sides has a, b, c, a, b and c adjacent tiles, correspondingly.

To better visualize the situation, look at the picture showing a similar hexagon for $a = 2$, $b = 3$ and $c = 4$.

According to the legend, as the King of Berland obtained the values a, b and c, he almost immediately calculated the total number of tiles on the hall floor. Can you do the same?

Input

The first line contains three integers: a, b and c ($2 \le a, b, c \le 1000$).

Output

Print a single number — the total number of tiles on the hall floor.

Examples

input
2 3 4
output
18

B. Forming Teams

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

One day n students come to the stadium. They want to play football, and for that they need to split into teams, the teams must have an equal number of people.

We know that this group of people has archenemies. Each student has at most two archenemies. Besides, if student A is an archenemy to student B, then student B is an archenemy to student A.

The students want to split so as no two archenemies were in one team. If splitting in the required manner is impossible, some students will have to sit on the bench.

Determine the minimum number of students you will have to send to the bench in order to form the two teams in the described manner and begin the game at last.

Input

The first line contains two integers n and m ($2 \le n \le 100$, $1 \le m \le 100$) — the number of students and the number of pairs of archenemies correspondingly.

Next m lines describe enmity between students. Each enmity is described as two numbers a_i and b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$) — the indexes of the students who are enemies to each other. Each enmity occurs in the list exactly once. It is guaranteed that each student has no more than two archenemies.

You can consider the students indexed in some manner with distinct integers from 1 to n.

Output

Print a single integer — the minimum number of students you will have to send to the bench in order to start the game.

Examples

input
5 4 1 2 2 4 5 3 1 4
output
1

input
6 2 1 4 3 4
output
0

input
6 6 1 2 2 3 3 1 4 5 5 6 6 4
output
2

C. Hiring Staff

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

A new Berland businessman Vitaly is going to open a household appliances' store. All he's got to do now is to hire the staff.

The store will work seven days a week, but not around the clock. Every day at least k people must work in the store.

Berland has a law that determines the order of working days and non-working days. Namely, each employee must work for exactly n consecutive days, then rest for exactly m days, then work for n more days and rest for m more, and so on. Vitaly doesn't want to break the law. Fortunately, there is a loophole: the law comes into force on the day when the employee is hired. For example, if an employee is hired on day x, then he should work on days $[x, x + 1, \ldots, x + n - 1]$, $[x + m + n, x + m + n + 1, \ldots, x + m + 2n - 1]$, and so on. Day x can be chosen arbitrarily by Vitaly.

There is one more thing: the key to the store. Berland law prohibits making copies of keys, so there is only one key. Vitaly is planning to entrust the key to the store employees. At the same time on each day the key must be with an employee who works that day — otherwise on this day no one can get inside the store. During the day the key holder can give the key to another employee, if he also works that day. The key will handed to the first hired employee at his first working day.

Each employee has to be paid salary. Therefore, Vitaly wants to hire as few employees as possible provided that the store can operate normally on each day from 1 to infinity. In other words, on each day with index from 1 to infinity, the store must have at least k working employees, and one of the working employees should have the key to the store.

Help Vitaly and determine the minimum required number of employees, as well as days on which they should be hired.

Input

The first line contains three integers n, m and k ($1 \le m \le n \le 1000$, $n \neq 1$, $1 \le k \le 1000$).

Output

In the first line print a single integer z — the minimum required number of employees.

In the second line print z positive integers, separated by spaces: the i-th integer a_i ($1 \le a_i \le 10^4$) should represent the number of the day, on which Vitaly should hire the i-th employee.

If there are multiple answers, print any of them.

Examples

input
4 3 2

output
4 1 1 4 5

input
3 3 1

output
3 1 3 5

D. Spider's Web

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

Paw the Spider is making a web. Web-making is a real art, Paw has been learning to do it his whole life. Let's consider the structure of the web.

There are n *main* threads going from the center of the web. All main threads are located in one plane and divide it into n equal infinite *sectors*. The sectors are indexed from 1 to n in the clockwise direction. Sectors i and $i + 1$ are *adjacent* for every i, $1 \le i < n$. In addition, sectors 1 and n are also adjacent.

Some sectors have *bridge threads*. Each bridge connects the two main threads that make up this sector. The points at which the bridge is attached to the main threads will be called *attachment points*. Both attachment points of a bridge are at the same distance from the center of the web. At each attachment point exactly one bridge is attached. The bridges are *adjacent* if they are in the same sector, and there are no other bridges between them.

A *cell* of the web is a trapezoid, which is located in one of the sectors and is bounded by two main threads and two adjacent bridges. You can see that the sides of the cell may have the attachment points of bridges from adjacent sectors. If the number of attachment points on one side of the cell is not equal to the number of attachment points on the other side, it creates an imbalance of pulling forces on this cell and this may eventually destroy the entire web. We'll call such a cell *unstable*. The perfect web does not contain unstable cells.

Unstable cells are marked red in the figure. Stable cells are marked green.

Paw the Spider isn't a skillful webmaker yet, he is only learning to make perfect webs. Help Paw to determine the number of unstable cells in the web he has just spun.

Input

The first line contains integer n ($3 \le n \le 1000$) — the number of main threads.

The i-th of following n lines describe the bridges located in the i-th sector: first it contains integer k_i ($1 \le k_i \le 10^5$) equal to the number of bridges in the given sector. Then follow k_i different integers p_{ij} ($1 \le p_{ij} \le 10^5$; $1 \le j \le k_i$). Number p_{ij} equals the distance from the attachment points of the j-th bridge of the i-th sector to the center of the web.

It is guaranteed that any two bridges between adjacent sectors are attached at a different distance from the center of the web. It is guaranteed that the total number of the bridges doesn't exceed 10^5.

Output

Print a single integer — the number of unstable cells in Paw the Spider's web.

Examples

input
7 3 1 6 7 4 3 5 2 9 2 8 1 4 3 7 6 4 3 2 5 9 3 6 3 8 3 4 2 9
output
6

E. Martian Luck

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

You know that the Martians use a number system with base k. Digit b ($0 \le b < k$) is considered *lucky*, as the first contact between the Martians and the Earthlings occurred in year b (by Martian chronology).

A *digital root* $d(x)$ of number x is a number that consists of a single digit, resulting after cascading summing of all digits of number x. Word "cascading" means that if the first summing gives us a number that consists of several digits, then we sum up all digits again, and again, until we get a one digit number.

For example, $d(3504_7) = d((3 + 5 + 0 + 4)_7) = d(15_7) = d((1 + 5)_7) = d(6_7) = 6_7$. In this sample the calculations are performed in the 7-base notation.

If a number's digital root equals b, the Martians also call this number lucky.

You have string s, which consists of n digits in the k-base notation system. Your task is to find, how many distinct substrings of the given string are lucky numbers. Leading zeroes are permitted in the numbers.

Note that substring $s[i... j]$ of the string $s = a_1a_2... a_n$ ($1 \le i \le j \le n$) is the string $a_ia_{i+1}... a_j$. Two substrings $s[i_1... j_1]$ and $s[i_2... j_2]$ of the string s are different if either $i_1 \ne i_2$ or $j_1 \ne j_2$.

Input

The first line contains three integers k, b and n ($2 \le k \le 10^9$, $0 \le b < k$, $1 \le n \le 10^5$).

The second line contains string s as a sequence of n integers, representing digits in the k-base notation: the i-th integer equals a_i ($0 \le a_i < k$) — the i-th digit of string s. The numbers in the lines are space-separated.

Output

Print a single integer — the number of substrings that are lucky numbers.

Please, do not use 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
10 5 6 3 2 0 5 6 1
output
5

input
7 6 4 3 5 0 4
output
1

input
257 0 3 0 0 256
output
3

Note

In the first sample the following substrings have the sought digital root: $s[1... 2]$ = "3 2", $s[1... 3]$ = "3 2 0", $s[3... 4]$ = "0 5", $s[4... 4]$ = "5" and $s[2... 6]$ = "2 0 5 6 1".