A. King Escape

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

Alice and Bob are playing chess on a huge chessboard with dimensions $n \times n$. Alice has a single piece left — a queen, located at (a_x, a_y), while Bob has only the king standing at (b_x, b_y). Alice thinks that as her queen is dominating the chessboard, victory is hers.

But Bob has made a devious plan to seize the victory for himself — he needs to march his king to (c_x, c_y) in order to claim the victory for himself. As Alice is distracted by her sense of superiority, **she no longer moves any pieces around, and it is only Bob who makes any turns.**

Bob will win if he can move his king from (b_x, b_y) to (c_x, c_y) **without ever getting in check**. Remember that a king can move to any of the 8 adjacent squares. A king is in check if it is on the same rank (i.e. row), file (i.e. column), or diagonal as the enemy queen.

Find whether Bob can win or not.

Input

The first line contains a single integer n ($3 \leq n \leq 1000$) — the dimensions of the chessboard.

The second line contains two integers a_x and a_y ($1 \leq a_x, a_y \leq n$) — the coordinates of Alice's queen.

The third line contains two integers b_x and b_y ($1 \leq b_x, b_y \leq n$) — the coordinates of Bob's king.

The fourth line contains two integers c_x and c_y ($1 \leq c_x, c_y \leq n$) — the coordinates of the location that Bob wants to get to.

It is guaranteed that Bob's king is currently not in check and the target location is not in check either.

Furthermore, the king is not located on the same square as the queen (i.e. $a_x \neq b_x$ or $a_y \neq b_y$), and the target does coincide neither with the queen's position (i.e. $c_x \neq a_x$ or $c_y \neq a_y$) nor with the king's position (i.e. $c_x \neq b_x$ or $c_y \neq b_y$).

Output

Print "YES" (without quotes) if Bob can get from (b_x, b_y) to (c_x, c_y) without ever getting in check, otherwise print "NO".

You can print each letter in any case (upper or lower).

Examples

input
8 4 4 1 3 3 1
output
YES

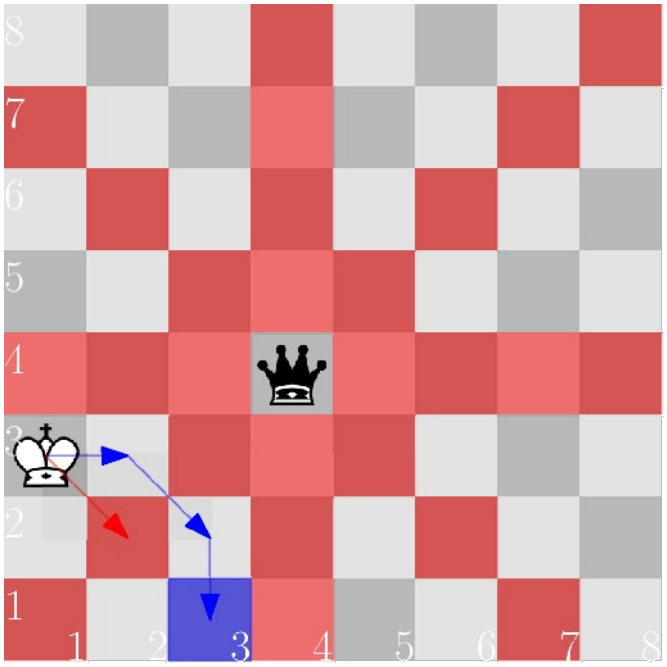
input
8 4 4 2 3 1 6
output
NO

input
8 3 5 1 2 6 1
output
NO

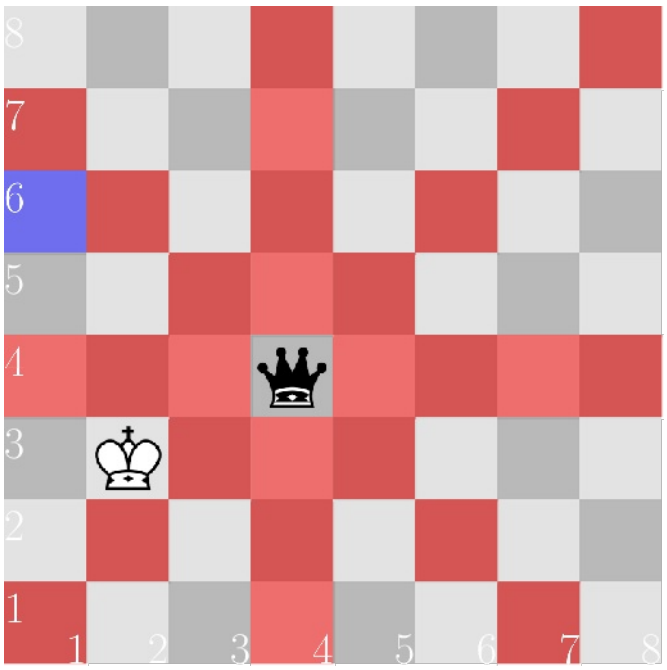
Note

In the diagrams below, the squares controlled by the black queen are marked red, and the target square is marked blue.

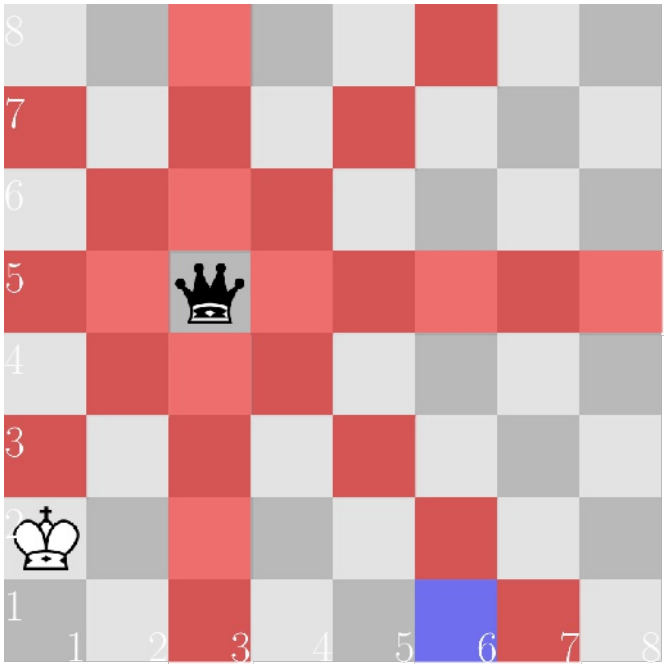
In the first case, the king can move, for instance, via the squares $(2, 3)$ and $(3, 2)$. Note that the direct route through $(2, 2)$ goes through check.



In the second case, the queen watches the fourth rank, and the king has no means of crossing it.



In the third case, the queen watches the third file.



B. Square Difference

time limit per test: 2 seconds
memory limit per test: 256 megabytes

input: standard input
output: standard output

Alice has a lovely piece of cloth. It has the shape of a **square** with a side of length a centimeters. Bob also wants such piece of cloth. He would prefer a **square** with a side of length b centimeters (where $b < a$). Alice wanted to make Bob happy, so she cut the needed square out of the corner of her piece and gave it to Bob. Now she is left with an ugly L shaped cloth (see pictures below).

Alice would like to know whether the area of her cloth expressed in square centimeters is prime. Could you help her to determine it?

Input
The first line contains a number t ($1 \le t \le 5$) — the number of test cases.

Each of the next t lines describes the i-th test case. It contains two integers a and b ($1 \le b < a \le 10^{11}$) — the side length of Alice's square and the side length of the square that Bob wants.

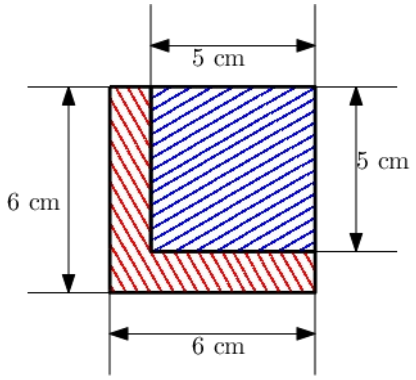
Output
Print t lines, where the i-th line is the answer to the i-th test case. Print "YES" (without quotes) if the area of the remaining piece of cloth is prime, otherwise print "NO".

You can print each letter in an arbitrary case (upper or lower).

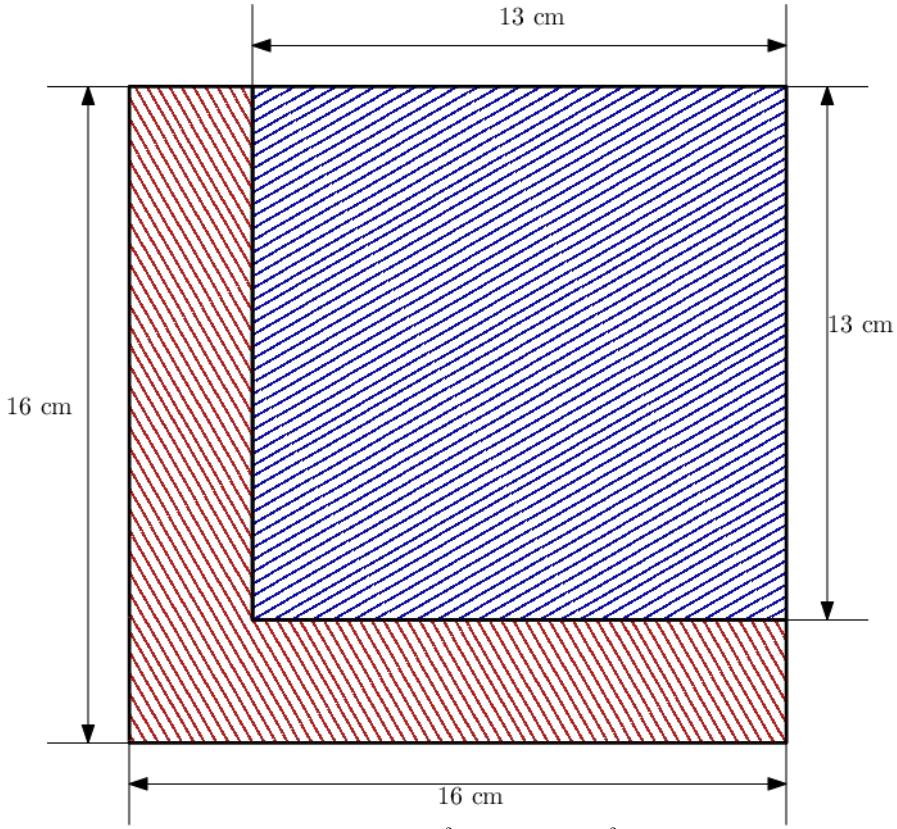
Example

input
4 6 5 16 13 61690850361 24777622630 34 33
output
YES NO NO YES

Note
The figure below depicts the first test case. The blue part corresponds to the piece which belongs to Bob, and the red part is the piece that Alice keeps for herself. The area of the red part is $6^2 - 5^2 = 36 - 25 = 11$, which is prime, so the answer is "YES".



In the second case, the area is $16^2 - 13^2 = 87$, which is divisible by 3.



In the third case, the area of the remaining piece is $61690850361^2 - 24777622630^2 = 3191830435068605713421$. This number is not prime because $3191830435068605713421 = 36913227731 \cdot 86468472991$.

In the last case, the area is $34^2 - 33^2 = 67$.

C. Permutation Game

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

After a long day, Alice and Bob decided to play a little game. The game board consists of n cells in a straight line, numbered from 1 to n, where each cell contains a number a_i between 1 and n. Furthermore, no two cells contain the same number.

A token is placed in one of the cells. They take alternating turns of moving the token around the board, with Alice moving first. The current player can move from cell i to cell j only if the following two conditions are satisfied:

- the number in the new cell j must be strictly larger than the number in the old cell i (i.e. $a_j > a_i$), and
- the distance that the token travels during this turn must be a multiple of the number in the old cell (i.e. $|i - j| \bmod a_i = 0$).

Whoever is unable to make a move, loses. For each possible starting position, determine who wins if they both play optimally. It can be shown that the game is always finite, i.e. there always is a winning strategy for one of the players.

Input
The first line contains a single integer n ($1 \le n \le 10^5$) — the number of numbers.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$). Furthermore, there are no pair of indices $i \ne j$ such that $a_i = a_j$.

Output
Print s — a string of n characters, where the i-th character represents the outcome of the game if the token is initially placed in the cell i. If Alice wins, then s_i has to be equal to "A"; otherwise, s_i has to be equal to "B".

Examples

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

input
15 3 11 2 5 10 9 7 13 15 8 4 12 6 1 14
output
ABAAAABBBAABAAB

Note
In the first sample, if Bob puts the token on the number (**not position**):

- 1: Alice can move to any number. She can win by picking 7, from which Bob has no move.
- 2: Alice can move to 3 and 5. Upon moving to 5, Bob can win by moving to 8. If she chooses 3 instead, she wins, as Bob has only a move to 4, from which Alice can move to 8.
- 3: Alice can only move to 4, after which Bob wins by moving to 8.
- 4, 5, or 6: Alice wins by moving to 8.
- 7, 8: Alice has no move, and hence she loses immediately.

D. Divisors

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

You are given n integers $a_1, a_2, \ldots, a_n$. Each of a_i has between 3 and 5 divisors. Consider $a = \prod a_i$ — the product of all input integers. Find the number of divisors of a. As this number may be very large, print it modulo prime number 998244353.

Input
The first line contains a single integer n ($1 \le n \le 500$) — the number of numbers.

Each of the next n lines contains an integer a_i ($1 \le a_i \le 2 \cdot 10^{18}$). It is guaranteed that the number of divisors of each a_i is between 3 and 5.

Output
Print a single integer d — the number of divisors of the product $a_1 \cdot a_2 \cdot \cdots \cdot a_n$ modulo 998244353.

Hacks input

For hacks, the input needs to be provided in a special format.

The first line contains an integer n ($1 \le n \le 500$) — the number of numbers.

Each of the next n lines contains a prime factorization of a_i. The line contains an integer k_i ($2 \le k_i \le 4$) — the number of prime factors of a_i and k_i integers $p_{i,j}$ ($2 \le p_{i,j} \le 2 \cdot 10^{18}$) where $p_{i,j}$ is the j-th prime factor of a_i.

Before supplying the input to the contestant, $a_i = \prod p_{i,j}$ are calculated. Note that each $p_{i,j}$ must be prime, each computed a_i must satisfy $a_i \le 2 \cdot 10^{18}$ and must have between 3 and 5 divisors. The contestant will be given only a_i, and not its prime factorization.

For example, you need to use this test to get the first sample:

2 3 3
2 3 5
2 11 13

Interaction

From the technical side, this problem is interactive. Therefore, do not forget to output end of line and flush the output. Also, do not read more than you need. To flush the output, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

Examples

input
3 9 15 143
output
32

input
1 7400840699802997
output
4

input
8 4606061759128693 4606066102679989 4606069767552943 4606063116488033 4606063930903637 4606064745319241 4606063930904021 4606065559735517
output
1920

input
3 4 8 16
output
10

Note

In the first case, $a = 19305$. Its divisors are $1, 3, 5, 9, 11, 13, 15, 27, 33, 39, 45, 55, 65, 99, 117, 135, 143, 165, 195, 297, 351, 429, 495, 585, 715, 1287, 1485, 1755, 2145, 3861, 6435, 19305$ — a total of 32.

In the second case, a has four divisors: 1, 86028121, 86028157, and 7400840699802997.

In the third case $a = 20260044567192536469873906162908387798196206970314026851657056488869937520947721404510225376602307240155749105445360$.

In the fourth case, $a = 512 = 2^9$, so answer equals to 10.

E. Hidden Bipartite Graph

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Bob has a simple undirected connected graph (without self-loops and multiple edges). He wants to learn whether his graph is bipartite (that is, you can paint all vertices of the graph into two colors so that there is no edge connecting two vertices of the same color) or not. As he is not very good at programming, he asked Alice for help. He does not want to disclose his graph to Alice, but he agreed that Alice can ask him some questions about the graph.

The only question that Alice can ask is the following: she sends s — a subset of vertices of the original graph. Bob answers with the number of edges that have both endpoints in s. Since he doesn't want Alice to learn too much about the graph, he allows her to ask no more than 20000 questions. Furthermore, he suspects that Alice might introduce false messages to their communication channel, so when Alice finally tells him whether the graph is bipartite or not, she also needs to provide a proof — either the partitions themselves or a cycle of odd length.

Your task is to help Alice to construct the queries, find whether the graph is bipartite.

Input

The first line contains a single integer n ($1 \le n \le 600$) — the number of vertices in Bob's graph.

Interaction

First, read an integer n ($1 \le n \le 600$) — the number of vertices in Bob's graph.

To make a query, print two lines. First of which should be in the format "? k" ($1 \le k \le n$), where k is the size of the set to be queried. The second line should contain k space separated distinct integers $s_1, s_2, \ldots, s_k$ ($1 \le s_i \le n$) — the vertices of the queried set.

After each query read a single integer m ($0 \le m \le \frac{n(n-1)}{2}$) — the number of edges between the vertices of the set $\{s_i\}$.

You are not allowed to ask more than 20000 queries.

If $m = -1$, it means that you asked more queries than allowed, or asked an invalid query. Your program should immediately terminate (for example, by calling exit(0)). You will receive `Wrong Answer`; it means that you asked more queries than allowed, or asked an invalid query. If you ignore this, you can get other verdicts since your program will continue to read from a closed stream.

After printing a query do not forget to print end of line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

When you know the answer, you need to print it.

The format of the answer depends on whether the graph is bipartite or not.

If the graph is bipartite, print two lines. The first should contain the letter "Y" (short for "YES") followed by a space, and then a single integer s ($0 \le s \le n$) — the number of vertices in one of the partitions. Second line should contain s integers $a_1, a_2, \ldots, a_s$ — vertices belonging to the first partition. All a_i must be distinct, and all edges in the main graph must have exactly one endpoint in the set $\{a_i\}$.

If the graph is not bipartite, print two lines. The first should contain the letter "N" (short for "NO") followed by a space, and then a single integer l ($3 \le l \le n$) — the length of one simple cycle of odd length. Second line should contain l integers $c_1, c_2, \ldots, c_l$ — the vertices along the cycle. It must hold that for all $1 \le i \le l$, there is an edge $\{c_i, c_{(i \bmod l)+1}\}$ in the main graph, and all c_i are distinct.

If there are multiple possible answers, you may print any of them.

Hacks format For hacks, use the following format:

The first line contains two integers n and m ($1 \le n \le 600, 0 \le m \le \frac{n(n-1)}{2}$) — the number of vertices and edges of the graph, respectively.

Each of the next m lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$) mean that there is an edge between u_i and v_i. There must not be any multiple edges, no loops, and the graph must be connected.

For example, you need to use this test to get the first sample:

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

Examples

input
4 4 0 1 1 1 0

output
? 4 1 2 3 4 ? 2 1 2 ? 2 1 3 ? 2 1 4 ? 2 2 4 ? 2 3 4 Y 2 1 2

input
4 4 3

output
? 4 1 4 2 3 ? 3 1 2 4 N 3 2 1 4

Note

In the first case, Alice learns that there are 4 edges in the whole graph. Over the course of the next three queries, she learns that vertex 1 has two neighbors: 3 and 4. She then learns that while vertex 2 is adjacent to 4, the vertex 3 isn't adjacent to 4. There is

only one option for the remaining edge, and that is $(2, 3)$. This means that the graph is a cycle on four vertices, with $(1, 2)$ being one partition and $(3, 4)$ being the second. Here, it would be also valid to output "3 4" on the second line.

In the second case, we also have a graph on four vertices and four edges. In the second query, Alice learns that there are three edges among vertices $(1, 2, 4)$. The only way this could possibly happen is that those form a triangle. As the triangle is not bipartite, Alice can report it as a proof. Notice that she does not learn where the fourth edge is, but she is able to answer Bob correctly anyway.

F. Boolean Computer

time limit per test: 7 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alice has a computer that operates on w-bit integers. The computer has n **registers** for values. The current content of the registers is given as an array $a_1, a_2, \ldots, a_n$.

The computer uses so-called "*number gates*" to manipulate this data. Each "*number gate*" takes two registers as inputs and calculates a function of the two values stored in those registers. Note that you can use the same register as both inputs.

Each "*number gate*" is assembled from bit gates. There are six types of bit gates: `AND`, `OR`, `XOR`, `NOT AND`, `NOT OR`, and `NOT XOR`, denoted "A", "O", "X", "a", "o", "x", respectively. Each **bit gate** takes two bits as input. Its output given the input bits b_1, b_2 is given below:

b_1	b_2	A	O	X	a	o	x
0	0	0	0	0	1	1	1
0	1	0	1	1	1	0	0
1	0	0	1	1	1	0	0
1	1	1	1	0	0	0	1

To build a "*number gate*", one takes w bit gates and assembles them into an array. A "*number gate*" takes two w-bit integers x_1 and x_2 as input. The "*number gate*" splits the integers into w bits and feeds the i-th bit of each input to the i-th bit gate. After that, it assembles the resulting bits again to form an output word.

For instance, for 4-bit computer we might have a "*number gate*" "AXoA" (`AND`, `XOR`, `NOT OR`, `AND`). For two inputs, $13 = 1101_2$ and $10 = 1010_2$, this returns $12 = 1100_2$, as 1 and 1 is 1, 1 `xor` 0 is 1, `not` (0 `or` 1) is 0, and finally 1 and 0 is 0.

You are given a description of m "*number gates*". For each gate, your goal is to report the number of register pairs for which the "*number gate*" outputs the number 0. In other words, find the number of ordered pairs (i, j) where $1 \le i, j \le n$, such that $w_k(a_i, a_j) = 0$, where w_k is the function computed by the k-th "*number gate*".

Input

The first line contains three integers: w, n, and m $(1 \le w \le 12, 1 \le n \le 3 \cdot 10^4, 1 \le m \le 5 \cdot 10^4)$ — the word size, the number of variables, and the number of gates.

The second line contains n integers $a_1, a_2, \ldots, a_n$ $(0 \le a_i < 2^w)$ — the value of variables stored in the registers.

Each of the next m lines contains a string g_j $(|g_j| = w)$ with a description of a single gate. Each character of g_j is one of "A", "O", "X", "a", "o", "x".

Output

Print m lines. The i-th line should contain the number of ordered pairs of variables for which the i-th gate returns zero.

Examples

input
4 3 1 13 10 6 AXoA
output
3

input
1 7 6 0 1 1 0 1 0 0 A O X a o x
output
40 16 25 9 33 24

input
6 2 4 47 12 AOXaox AAaaAA xxxxxx XXXXXX
output
2 3 0 2

input
2 2 2 2 0 xO Ox

output
2 0

Note

In the first test case, the inputs in binary are 1101, 1010, 0110. The pairs that return 0 are $(13, 6)$, $(6, 13)$, and $(6, 6)$. As it was already mentioned in the problem statement, $13 \oplus 10 = 10 \oplus 13 = 12$. The other pairs are $13 \oplus 13 = 11$, $10 \oplus 10 = 8$ and $10 \oplus 6 = 6 \oplus 10 = 4$.

G. Chip Game

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alice and Bob decided to play one ultimate game. They have n piles, the i-th pile initially contain v_i chips. Alice selects a positive integer a from interval $[1, m]$, and Bob selects a number b the same way.

Then the game starts. In her turn, Alice can select any pile containing at least a chips, and remove exactly a chips from it. Similarly, in his turn, Bob can choose any pile with at least b chips, and remove exactly b chips from it. If a player cannot make a move, he or she loses.

If both players play optimally, the outcome ultimately depends on the choice of a and b, and on the starting player. Consider a fixed pair (a, b). There are four types of games:

- **Alice** wins, regardless of who starts.
- **Bob** wins, regardless of who starts.
- If Alice starts, she wins. If Bob starts, he wins. We say that the **first** player wins.
- If Alice starts, Bob wins. If Bob starts, Alice wins. We say that the **second** player wins.

Among all choices of a and b (i.e. for each pair (a, b) such that $1 \leq a, b \leq m$), determine how many games are won by Alice (regardless of who starts), how many are won by Bob (regardless of who starts), how many are won by the first player, and how many are won by the second player.

Input

The first line contains two integers n and m ($1 \leq n \leq 100, 1 \leq m \leq 10^5$) — the number of piles, and the upper bound on the number of chips allowed to be taken in one turn, respectively.

The second line contains n integers $v_1, v_2, \ldots, v_n$ ($1 \leq v_i \leq 10^{18}$) — the starting number of chips in each pile.

Output

Print a single line containing four integers w_a, w_b, w_f, w_s — the number of games won by Alice, Bob, the first player, the second player, respectively.

Examples

input
2 2 4 5

output
1 1 1 1

input
2 20 4 5

output
82 82 6 230

Note

In the first sample, there are a total of four choices for the tuple (a, b).

- $a = b = 1$: It does not matter from which pile the chips are removed — there are 9 turns to be made, and the first player also makes the last, and hence he wins.
- $a = b = 2$: The second player may win by always selecting the same pile as the first before there are 0 and 1 chips in the piles. Since it is impossible to remove 2 chips from any of them, the first player loses.
- $a = 1$ and $b = 2$:
  - Suppose Alice starts. She can win by removing a single chip from the pile with 5 chips and then copying Bob's decision. After the next four turns, the game ends with 1 chip in each pile and Bob unable to make a move.
  - Suppose Bob starts. If he removes two chips from second pile, Alice counters by removing a chip from the first pile. From then on, she can always copy Bob's decision and win. If Bob removes two chips from the first pile in his first turn, Alice can also remove one. As the first pile only has one chip left, Bob is forced to remove two chips from the second pile, leaving 3. Regardless of what Alice does, she wins.

  Hence, Alice wins no matter who starts.
- $a = 2$ and $b = 1$: This is symmetric with the previous case — hence Bob always wins.

In the second sample, a lot of games, e.g. $a = 7$ and $b = 6$, end without anybody being able to remove single chip — which counts as a win for the second player. The games won for the first player are $(1, 1)$, $(1, 4)$, $(2, 3)$, $(3, 2)$, $(4, 1)$, and $(5, 5)$.

The only programming contests Web 2.0 platform