A. Boboniu Likes to Color Balls

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

Boboniu gives you

- r red balls,
- g green balls,
- b blue balls,
- w white balls.

He allows you to do the following operation as many times as you want:

- Pick a red ball, a green ball, and a blue ball and then change their color to white.

You should answer if it's possible to arrange all the balls into a *palindrome* after several (possibly zero) number of described operations.

Input

The first line contains one integer T ($1 \le T \le 100$) denoting the number of test cases.

For each of the next T cases, the first line contains four integers r, g, b and w ($0 \le r, g, b, w \le 10^9$).

Output

For each test case, print "Yes" if it's possible to arrange all the balls into a palindrome after doing several (possibly zero) number of described operations. Otherwise, print "No".

Example

input

```
4
0 1 1 1
8 1 9 3
0 0 0 0
1000000000 1000000000 1000000000 1000000000
```

output

```
No
Yes
Yes
Yes
```

Note

In the first test case, you're not able to do any operation and you can never arrange three balls of distinct colors into a palindrome.

In the second test case, after doing one operation, changing $(8,1,9,3)$ to $(7,0,8,6)$, one of those possible palindromes may be "rrrwwwbbbbrbbbbwwwrrr".

A *palindrome* is a word, phrase, or sequence that reads the same backwards as forwards. For example, "rggbwbggr", "b", "gg" are palindromes while "rgbb", "gbbgr" are not. Notice that an **empty** word, phrase, or sequence is palindrome.

B. Boboniu Plays Chess

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

Boboniu likes playing chess with his employees. As we know, no employee can beat the boss in the chess game, so Boboniu has never lost in any round.

You are a new applicant for his company. Boboniu will test you with the following chess question:

Consider a $n \times m$ grid (rows are numbered from 1 to n, and columns are numbered from 1 to m). You have a chess piece, and it stands at some cell (S_x, S_y) which is not on the border (i.e. $2 \le S_x \le n-1$ and $2 \le S_y \le m-1$).

From the cell (x, y), you can move your chess piece to (x, y') ($1 \le y' \le m, y' \neq y$) or (x', y) ($1 \le x' \le n, x' \neq x$). In other words, the chess piece moves as a rook. From the cell, you can move to any cell on the same row or column.

Your goal is to visit each cell exactly once. Can you find a solution?

Note that cells on the path between two adjacent cells in your route are not counted as visited, and it is not required to return to the starting point.

Input

The only line of the input contains four integers n, m, S_x and S_y ($3 \le n, m \le 100$, $2 \le S_x \le n-1$, $2 \le S_y \le m-1$) — the number of rows, the number of columns, and the initial position of your chess piece, respectively.

Output

You should print $n \cdot m$ lines.

The i-th line should contain two integers x_i and y_i ($1 \le x_i \le n$, $1 \le y_i \le m$), denoting the i-th cell that you visited. You should print exactly nm pairs (x_i, y_i), they should cover all possible pairs (x_i, y_i), such that $1 \le x_i \le n$, $1 \le y_i \le m$.

We can show that under these constraints there always exists a solution. If there are multiple answers, print any.

Examples

input

```
3 3 2 2
```

output

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

input

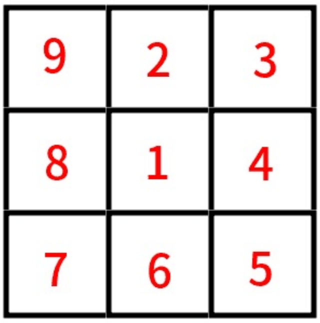
```
3 4 2 2
```

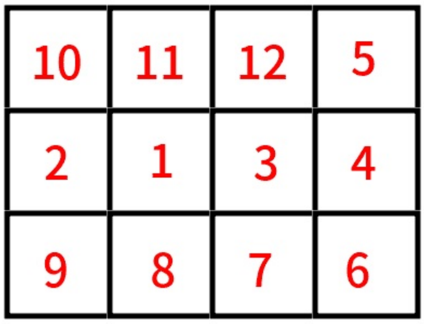
output

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

Note

Possible routes for two examples:





C. Boboniu and Bit Operations

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

Boboniu likes bit operations. He wants to play a game with you.

Boboniu gives you two sequences of non-negative integers $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_m$.

For each i ($1 \le i \le n$), you're asked to choose a j ($1 \le j \le m$) and let $c_i = a_i \& b_j$, where $\&$ denotes the bitwise AND operation. Note that you can pick the same j for different i's.

Find the minimum possible $c_1 | c_2 | \ldots | c_n$, where $|$ denotes the bitwise OR operation.

Input

The first line contains two integers n and m ($1 \le n, m \le 200$).

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i < 2^9$).

The next line contains m integers $b_1, b_2, \ldots, b_m$ ($0 \le b_i < 2^9$).

Output

Print one integer: the minimum possible $c_1 | c_2 | \ldots | c_n$.

Examples

input

```
4 2
2 6 4 0
2 4
```

output

```
2
```

input

```
7 6
1 9 1 9 8 1 0
1 1 4 5 1 4
```

output

```
0
```

input

```
8 5
179 261 432 162 82 43 10 38
379 357 202 184 197
```

output

```
147
```

Note

For the first example, we have $c_1 = a_1 \& b_2 = 0$, $c_2 = a_2 \& b_1 = 2$, $c_3 = a_3 \& b_1 = 0$, $c_4 = a_4 \& b_1 = 0$.Thus $c_1 | c_2 | c_3 | c_4 = 2$, and this is the minimal answer we can get.

D. Boboniu Chats with Du

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

Have you ever used the chat application QQ? Well, in a chat group of QQ, administrators can muzzle a user for days.

In Boboniu's chat group, there's a person called Du Yi who likes to make fun of Boboniu every day.

Du will chat in the group for n days. On the i-th day:

- If Du can speak, he'll make fun of Boboniu with fun factor a_i. But after that, he may be muzzled depending on Boboniu's mood.
- Otherwise, Du won't do anything.

Boboniu's mood is a constant m. On the i-th day:

- If Du can speak and $a_i > m$, then Boboniu will be angry and muzzle him for d days, which means that Du won't be able to speak on the $i+1, i+2, \cdots, \min(i+d, n)$-th days.
- Otherwise, Boboniu won't do anything.

The total fun factor is the sum of the fun factors on the days when Du can speak.

Du asked you to find the maximum total fun factor among all possible permutations of a.

Input

The first line contains three integers n, d and m ($1 \le d \le n \le 10^5, 0 \le m \le 10^9$).

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$).

Output

Print one integer: the maximum total fun factor among all permutations of a.

Examples

input

```
5 2 11
8 10 15 23 5
```

output

```
48
```

input

```
20 2 16
20 5 8 2 18 16 2 16 16 1 5 16 2 13 6 16 4 17 21 7
```

output

```
195
```

Note

In the first example, you can set $a' = [15, 5, 8, 10, 23]$. Then Du's chatting record will be:

1. Make fun of Boboniu with fun factor 15.
2. Be muzzled.
3. Be muzzled.
4. Make fun of Boboniu with fun factor 10.
5. Make fun of Boboniu with fun factor 23.

Thus the total fun factor is 48.

E. Boboniu Walks on Graph

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

Boboniu has a **directed** graph with n vertices and m edges.

The out-degree of each vertex is at most k.

Each edge has an integer weight between 1 and m. No two edges have equal weights.

Boboniu likes to walk on the graph with some specific rules, which is represented by a tuple $(c_1, c_2, \ldots, c_k)$. If he now stands on a vertex u with out-degree i, then he will go to the next vertex by the edge with the c_i-th $(1 \le c_i \le i)$ smallest weight among all edges outgoing from u.

Now Boboniu asks you to calculate the number of tuples $(c_1, c_2, \ldots, c_k)$ such that

- $1 \le c_i \le i$ for all i ($1 \le i \le k$).
- Starting from any vertex u, it is possible to go back to u in finite time by walking on the graph under the described rules.

Input

The first line contains three integers n, m and k ($2 \le n \le 2 \cdot 10^5, 2 \le m \le \min(2 \cdot 10^5, n(n-1)), 1 \le k \le 9$).

Each of the next m lines contains three integers u, v and w $(1 \le u, v \le n, u \ne v, 1 \le w \le m)$, denoting an edge from u to v with weight w. It is guaranteed that there are no self-loops or multiple edges and each vertex has at least one edge starting from itself.

It is guaranteed that the out-degree of each vertex is at most k and no two edges have equal weight.

Output

Print one integer: the number of tuples.

Examples

input

```
4 6 3
4 2 1
1 2 2
2 4 3
4 1 4
4 3 5
3 1 6
```

output

```
2
```

input

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

output

```
1
```

input

```
6 13 4
3 5 1
2 5 2
6 3 3
1 4 4
2 6 5
5 3 6
4 1 7
4 3 8
```

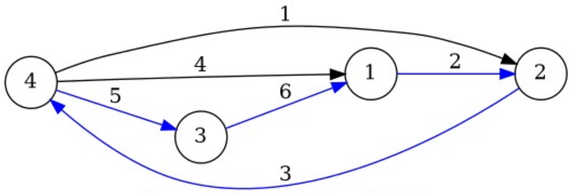
```
5 2 9
4 2 10
2 1 11
6 1 12
4 6 13
```

output

```
1
```

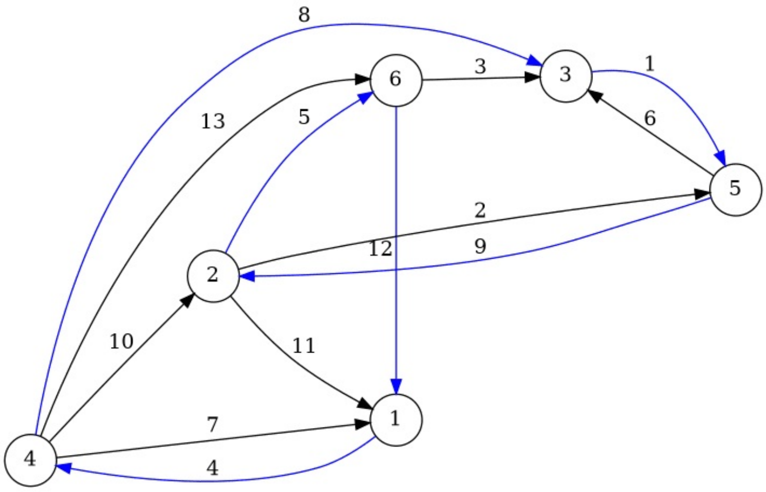
Note

For the first example, there are two tuples: $(1, 1, 3)$ and $(1, 2, 3)$. The blue edges in the picture denote the c_i-th smallest edges for each vertex, which Boboniu chooses to go through.



Example 1: (1,1,3) or (1,2,3)

For the third example, there's only one tuple: $(1, 2, 2, 2)$.



Example 3: (1,2,2,2)

The *out-degree* of vertex u means the number of edges outgoing from u.

F. Boboniu and String

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Boboniu defines *BN-string* as a string s of characters 'B' and 'N'.

You can perform the following operations on the BN-string s:

- Remove a character of s.
- Remove a substring "BN" or "NB" of s.
- Add a character 'B' or 'N' to the end of s.
- Add a string "BN" or "NB" to the end of s.

Note that a string a is a *substring* of a string b if a can be obtained from b by deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

Boboniu thinks that BN-strings s and t are *similar* if and only if:

- $|s| = |t|$.
- There exists a permutation $p_1, p_2, \ldots, p_{|s|}$ such that for all i ($1 \le i \le |s|$), $s_{p_i} = t_i$.

Boboniu also defines $\mathrm{dist}(s, t)$, the *distance* between s and t, as the minimum number of operations that makes s *similar* to t.

Now Boboniu gives you n non-empty BN-strings $s_1, s_2, \ldots, s_n$ and asks you to find a **non-empty** BN-string t such that the maximum distance to string s is minimized, i.e. you need to minimize $\max_{i=1}^{n} \mathrm{dist}(s_i, t)$.

Input

The first line contains a single integer n ($1 \le n \le 3 \cdot 10^5$).

Each of the next n lines contains a string s_i ($1 \le |s_i| \le 5 \cdot 10^5$). It is guaranteed that s_i only contains 'B' and 'N'. The sum of $|s_i|$ does not exceed $5 \cdot 10^5$.

Output

In the first line, print the minimum $\max_{i=1}^{n} \mathrm{dist}(s_i, t)$.

In the second line, print the suitable t.

If there are several possible t's, you can print any.

Examples

input

```
3
B
N
BN
```

output

```
1
BN
```

input

```
10
N
```

```
BBBBBB
BNNNBBNBB
NNNNBNBNNBNNNBBN
NBNBN
NNNNNN
BNBNBNBBBBNNNNBBBBNNBBNBNBBNBBBBBBBB
NNNNBN
NBBBBBBBB
NNNNNN
```

output

```
12
BBBBBBBBBBBBNNNNNNNNNNNN
```

input

```
8
NNN
NNN
BBNNBBBN
NNNBNN
B
NNN
NNNNBNN
NNNNNNNNNNNNNNNBNNNNNNNBNB
```

output

```
12
BBBBNNNNNNNNNNNN
```

input

```
3
BNNNBNNNNNBNBBNNNBBNNNNNBBBBNNBBBBBBNBBBBBNBBBNNBBBNBNBBBN
BBBNBBBBNNNNNBBNBBBNNNBB
BBBBBBBBBBBBBBNBBBBBNBBBBBNBBBBNB
```

output

```
12
BBBBBBBBBBBBBBBBBBBBBBBBBBNNNNNNNNNNNN
```

Note

In the first example $\mathrm{dist}(\mathrm{B},\mathrm{BN}) = \mathrm{dist}(\mathrm{N},\mathrm{BN}) = 1$, $\mathrm{dist}(\mathrm{BN},\mathrm{BN}) = 0$. So the maximum distance is 1.