A. Almost Arithmetical Progression

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

Gena loves sequences of numbers. Recently, he has discovered a new type of sequences which he called an almost arithmetical progression. A sequence is an *almost arithmetical progression*, if its elements can be represented as:

- $a_1 = p$, where p is some integer;
- $a_i = a_{i-1} + (-1)^{i+1} \cdot q$ $(i > 1)$, where q is some integer.

Right now Gena has a piece of paper with sequence b, consisting of n integers. Help Gena, find there the longest subsequence of integers that is an almost arithmetical progression.

Sequence $s_1, s_2, \ldots, s_k$ is a subsequence of sequence $b_1, b_2, \ldots, b_n$, if there is such increasing sequence of indexes $i_1, i_2, \ldots, i_k$ $(1 \le i_1 < i_2 < \ldots < i_k \le n)$, that $b_{i_j} = s_j$. In other words, sequence s can be obtained from b by crossing out some elements.

Input

The first line contains integer n $(1 \le n \le 4000)$. The next line contains n integers $b_1, b_2, \ldots, b_n$ $(1 \le b_i \le 10^6)$.

Output

Print a single integer — the length of the required longest subsequence.

Sample test(s)

input

```
2
3 5
```

output

```
2
```

input

```
4
10 20 10 30
```

output

```
3
```

Note

In the first test the sequence actually is the suitable subsequence.

In the second test the following subsequence fits: $10, 20, 10$.

B. Mr. Bender and Square

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

Mr. Bender has a digital table of size $n \times n$, each cell can be switched on or off. He wants the field to have at least c switched on squares. When this condition is fulfilled, Mr Bender will be happy.

We'll consider the table rows numbered from top to bottom from 1 to n, and the columns — numbered from left to right from 1 to n. Initially there is exactly one switched on cell with coordinates (x, y) (x is the row number, y is the column number), and all other cells are switched off. Then each second we switch on the cells that are off but have the side-adjacent cells that are on.

For a cell with coordinates (x, y) the side-adjacent cells are cells with coordinates $(x - 1, y)$, $(x + 1, y)$, $(x, y - 1)$, $(x, y + 1)$.

In how many seconds will Mr. Bender get happy?

Input

The first line contains four space-separated integers n, x, y, c $(1 \le n, c \le 10^9;\ 1 \le x, y \le n;\ c \le n^2)$.

Output

In a single line print a single integer — the answer to the problem.

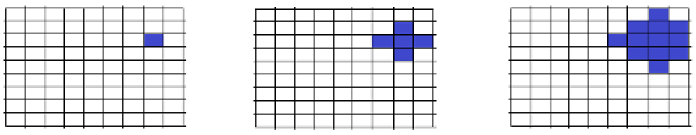
Sample test(s)

```
input
6 4 3 1
output
0
```

```
input
9 3 8 10
output
2
```

Note

Initially the first test has one painted cell, so the answer is 0. In the second test all events will go as is shown on the figure.



.

C. Furlo and Rublo and Game

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

Furlo and Rublo play a game. The table has n piles of coins lying on it, the i-th pile has a_i coins. Furlo and Rublo move in turns, Furlo moves first. In one move you are allowed to:

- choose some pile, let's denote the current number of coins in it as x;
- choose some integer y $(0 \le y < x; x^{1/4} \le y \le x^{1/2})$ and decrease the number of coins in this pile to y. In other words, after the described move the pile will have y coins left.

The player who can't make a move, loses.

Your task is to find out, who wins in the given game if both Furlo and Rublo play optimally well.

Input

The first line contains integer n $(1 \le n \le 77777)$ — the number of piles. The next line contains n integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le 777777777777)$ — the sizes of piles. The numbers are separated by single spaces.

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.

Output

If both players play optimally well and Furlo wins, print "`Furlo`", otherwise print "`Rublo`". Print the answers without the quotes.

Sample test(s)

input

```
1
1
```

output

```
Rublo
```

input

```
2
1 2
```

output

```
Rublo
```

input

```
10
1 2 3 4 5 6 7 8 9 10
```

output

```
Furlo
```

D. Liars and Serge

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

There are n people, sitting in a line at the table. For each person we know that he always tells either the truth or lies.

Little Serge asked them: how many of you always tell the truth? Each of the people at the table knows everything (who is an honest person and who is a liar) about all the people at the table. The honest people are going to say the correct answer, the liars are going to say any integer from 1 to n, which is not the correct answer. Every liar chooses his answer, regardless of the other liars, so two distinct liars may give distinct answer.

Serge does not know any information about the people besides their answers to his question. He took a piece of paper and wrote n integers $a_1, a_2, ..., a_n$, where a_i is the answer of the i-th person in the row. Given this sequence, Serge determined that exactly k people sitting at the table **apparently lie**.

Serge wonders, how many variants of people's answers (sequences of answers a of length n) there are where one can say that exactly k people sitting at the table apparently lie. As there can be rather many described variants of answers, count the remainder of dividing the number of the variants by 777777777.

Input

The first line contains two integers n, k, $(1 \le k \le n \le 2^8)$. It is guaranteed that n — is the power of number 2.

Output

Print a single integer — the answer to the problem modulo 777777777.

Sample test(s)

input

```
1 1
```

output

```
0
```

input

```
2 1
```

output

```
2
```

E. Lucky Arrays

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

Little Maxim loves interesting problems. He decided to share one such problem with you.

Initially there is an array a, consisting of n zeroes. The elements of the array are indexed, starting from 1. Then follow queries to change array a. Each query is characterized by two integers v_i, t_i. In the answer to the query we should make the v_i-th array element equal t_i $(a_{v_i} = t_i;\ 1 \le v_i \le n)$.

Maxim thinks that some pairs of integers (x, y) are good and some are not. Maxim thinks that array a, consisting of n integers, is lucky, if for all integer i, $(1 \le i \le n - 1)$ the pair of integers (a_i, a_{i+1}) — is good. Note that the order of numbers in the pairs is important, that is, specifically, $(1, 2) \ne (2, 1)$.

After each query to change array a Maxim wants to know, how many ways there are to replace all zeroes in array a with integers from one to three so as to make the resulting array (without zeroes) lucky. Of course, distinct zeroes can be replaced by distinct integers.

Maxim told you the sequence of queries and all pairs of integers he considers lucky. Help Maxim, solve this problem for him.

Input

The first line contains integers n and m $(1 \le n, m \le 77777)$ — the number of elements in the array and the number of commands.

The next three lines contain matrix w, consisting only of zeroes and ones; the j-th number in the i-th of these lines — $w_{i,j}$. If $w_{i,j} = 1$ $(1 \le i, j \le 3)$, then pair (i, j) is good, otherwise it is not good. Matrix does not have to be symmetric relative to the main diagonal.

Next m lines contain pairs of integers v_i, t_i $(1 \le v_i \le n, 0 \le t_i \le 3)$ — the queries to change the array.

Output

Print m integers — the i-th number should equal to the number of ways to replace all zeroes in array a (changed after the i-th query) by integers from one to three so as to make the resulting array (without zeroes) lucky. Separate the numbers by whitespaces. As the answers can be rather large, print the remainder from dividing them by 777777777.

Sample test(s)

input

```
3 10
1 1 0
1 0 0
1 1 1
1 1
1 3
2 2
3 0
2 1
3 0
3 1
2 0
3 1
1 0
```

output

```
3
6
1
1
2
2
1
3
3
6
```