Codeforces Round #431 (Div. 1)

A. From Y to Y

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

From beginning till end, this message has been waiting to be conveyed.

For a given unordered multiset of n lowercase English letters ("multi" means that a letter may appear more than once), we treat all letters as strings of length 1, and repeat the following operation n - 1 times:

- Remove any two elements s and t from the set, and add their concatenation $s + t$ to the set.

The cost of such operation is defined to be $\sum_{c \in \{\text{'a'}, \text{'b'}, \ldots, \text{'z'}\}} f(s, c) \cdot f(t, c)$, where $f(s, c)$ denotes the number of times character c appears in string s.

Given a non-negative integer k, construct any valid non-empty set of no more than $100\,000$ letters, such that the minimum accumulative cost of the whole process is **exactly** k. It can be shown that a solution always exists.

Input

The first and only line of input contains a non-negative integer k ($0 \le k \le 100\,000$) — the required minimum cost.

Output

Output a non-empty string of no more than $100\,000$ lowercase English letters — any multiset satisfying the requirements, concatenated to be a string.

Note that the printed string doesn't need to be the final concatenated string. It only needs to represent an unordered multiset of letters.

Examples

input
12
output
abababab

input
3
output
codeforces

Note

For the multiset {'a', 'b', 'a', 'b', 'a', 'b', 'a', 'b'}, one of the ways to complete the process is as follows:

- {"ab", "a", "b", "a", "b", "a", "b"}, with a cost of 0;
- {"aba", "b", "a", "b", "a", "b"}, with a cost of 1;
- {"abab", "a", "b", "a", "b"}, with a cost of 1;
- {"abab", "ab", "a", "b"}, with a cost of 0;
- {"abab", "aba", "b"}, with a cost of 1;
- {"abab", "abab"}, with a cost of 1;
- {"abababab"}, with a cost of 8.

The total cost is 12, and it can be proved to be the minimum cost of the process.

B. Rooter's Song

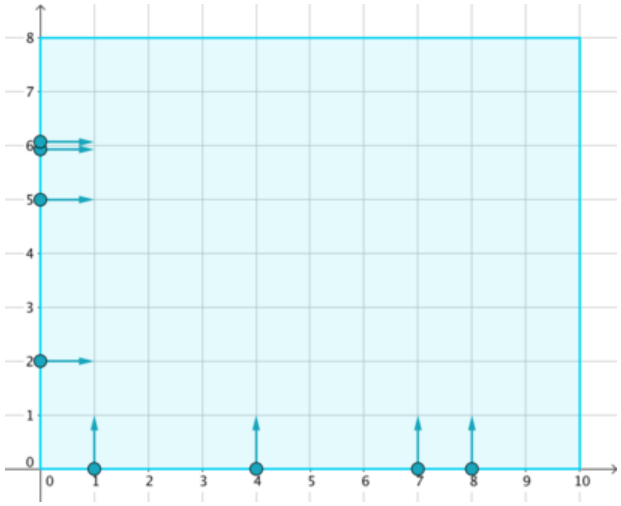
time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Wherever the destination is, whoever we meet, let's render this song together.

On a Cartesian coordinate plane lies a rectangular stage of size $w \times h$, represented by a rectangle with corners $(0, 0)$, $(w, 0)$, (w, h) and $(0, h)$. It can be seen that no collisions will happen before one enters the stage.

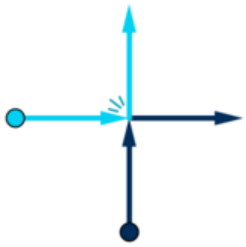
On the sides of the stage stand n dancers. The i-th of them falls into one of the following groups:

- **Vertical**: stands at $(x_i, 0)$, moves in positive y direction (upwards);
- **Horizontal**: stands at $(0, y_i)$, moves in positive x direction (rightwards).



According to choreography, the i-th dancer should stand still for the first t_i milliseconds, and then start moving in the specified direction at 1 unit per millisecond, until another border is reached. It is guaranteed that no two dancers have the same group, position and waiting time at the same time.

When two dancers collide (i.e. are on the same point at some time when both of them are moving), they immediately exchange their moving directions and go on.



Dancers stop when a border of the stage is reached. Find out every dancer's stopping position.

Input

The first line of input contains three space-separated positive integers n, w and h ($1 \le n \le 100\,000$, $2 \le w, h \le 100\,000$) — the number of dancers and the width and height of the stage, respectively.

The following n lines each describes a dancer: the i-th among them contains three space-separated integers g_i, p_i, and t_i ($1 \le g_i \le 2$, $1 \le p_i \le 99\,999$, $0 \le t_i \le 100\,000$), describing a dancer's group g_i ($g_i = 1$ — vertical, $g_i = 2$ — horizontal), position, and waiting time. If $g_i = 1$ then $p_i = x_i$; otherwise $p_i = y_i$. It's guaranteed that $1 \le x_i \le w - 1$ and $1 \le y_i \le h - 1$. It is guaranteed that no two dancers have the same group, position and waiting time at the same time.

Output

Output n lines, the i-th of which contains two space-separated integers (x_i, y_i) — the stopping position of the i-th dancer in the input.

Examples

input
8 10 8 1 1 10 1 4 13 1 7 1 1 8 2 2 2 0 2 5 14 2 6 0 2 6 1
output

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

input

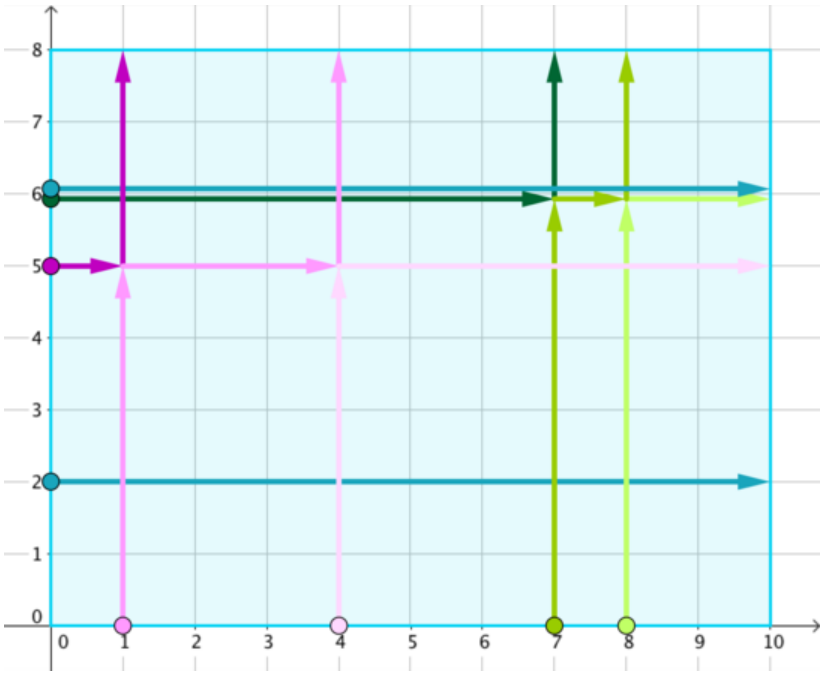
```
3 2 3
1 1 2
2 1 1
1 1 5
```

output

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

Note

The first example corresponds to the initial setup in the legend, and the tracks of dancers are marked with different colours in the following figure.



In the second example, no dancers collide.

C. Goodbye Souvenir

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

I won't feel lonely, nor will I be sorrowful... not before everything is buried.

A string of n beads is left as the message of leaving. The beads are numbered from 1 to n from left to right, each having a shape numbered by integers between 1 and n inclusive. Some beads may have the same shapes.

The *memory* of a shape x in a certain subsegment of beads, is defined to be the difference between the last position and the first position that shape x appears in the segment. The *memory* of a subsegment is the sum of *memories* over all shapes that occur in it.

From time to time, shapes of beads change as well as the *memories*. Sometimes, the past secreted in subsegments are being recalled, and you are to find the *memory* for each of them.

Input

The first line of input contains two space-separated integers n and m ($1 \le n, m \le 100\,000$) — the number of beads in the string, and the total number of changes and queries, respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the initial shapes of beads $1, 2, \ldots, n$, respectively.

The following m lines each describes either a change in the beads or a query of subsegment. A line has one of the following formats:

- `1 p x` ($1 \le p \le n$, $1 \le x \le n$), meaning that the shape of the p-th bead is changed into x;
- `2 l r` ($1 \le l \le r \le n$), denoting a query of *memory* of the subsegment from l to r, inclusive.

Output

For each query, print one line with an integer — the *memory* of the recalled subsegment.

Examples

input
7 6 1 2 3 1 3 2 1 2 3 7 2 1 3 1 7 2 1 3 2 2 1 6 2 5 7

output
5 0 7 1

input
7 5 1 3 2 1 4 2 3 1 1 4 2 2 3 1 1 7 2 4 5 1 1 7

output
0 0

Note

The initial string of beads has shapes $(1, 2, 3, 1, 3, 2, 1)$.

Consider the changes and queries in their order:

1. `2 3 7`: the *memory* of the subsegment $[3, 7]$ is $(7 - 4) + (6 - 6) + (5 - 3) = 5$;
2. `2 1 3`: the *memory* of the subsegment $[1, 3]$ is $(1 - 1) + (2 - 2) + (3 - 3) = 0$;
3. `1 7 2`: the shape of the 7-th bead changes into 2. Beads now have shapes $(1, 2, 3, 1, 3, 2, 2)$ respectively;
4. `1 3 2`: the shape of the 3-rd bead changes into 2. Beads now have shapes $(1, 2, 2, 1, 3, 2, 2)$ respectively;
5. `2 1 6`: the *memory* of the subsegment $[1, 6]$ is $(4 - 1) + (6 - 2) + (5 - 5) = 7$;
6. `2 5 7`: the *memory* of the subsegment $[5, 7]$ is $(7 - 6) + (5 - 5) = 1$.

D. Shake It!

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

A never-ending, fast-changing and dream-like world unfolds, as the secret door opens.

A *world* is an unordered graph G, in whose vertex set $V(G)$ there are two special vertices $s(G)$ and $t(G)$. An *initial world* has vertex set $\{s(G), t(G)\}$ and an edge between them.

A total of n changes took place in an *initial world*. In each change, a new vertex w is added into $V(G)$, **an existing edge** (u, v) is chosen, and two edges (u, w) and (v, w) are added into $E(G)$. Note that it's possible that some edges are chosen in more than one change.

It's known that the capacity of the minimum s-t cut of the resulting graph is m, that is, at least m edges need to be removed in order to make $s(G)$ and $t(G)$ disconnected.

Count the number of *non-similar worlds* that can be built under the constraints, modulo $10^9 + 7$. We define two *worlds* similar, if they are isomorphic and there is isomorphism in which the s and t vertices are not relabelled. Formally, two *worlds* G and H are considered similar, if there is a bijection between their vertex sets $f : V(G) \to V(H)$, such that:

- $f(s(G)) = s(H)$;
- $f(t(G)) = t(H)$;
- Two vertices u and v of G are adjacent in G if and only if $f(u)$ and $f(v)$ are adjacent in H.

Input

The first and only line of input contains two space-separated integers n, m ($1 \le n, m \le 50$) — the number of operations performed and the minimum cut, respectively.

Output

Output one integer — the number of *non-similar worlds* that can be built, modulo $10^9 + 7$.

Examples

input
3 2
output
6

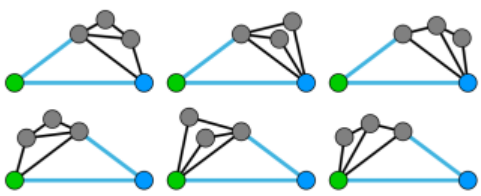
input
4 4
output
3

input
7 3
output
1196

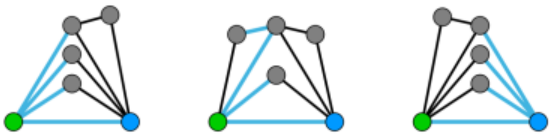
input
31 8
output
64921457

Note

In the first example, the following 6 *worlds* are pairwise non-similar and satisfy the constraints, with $s(G)$ marked in green, $t(G)$ marked in blue, and one of their minimum cuts in light blue.



In the second example, the following 3 *worlds* satisfy the constraints.



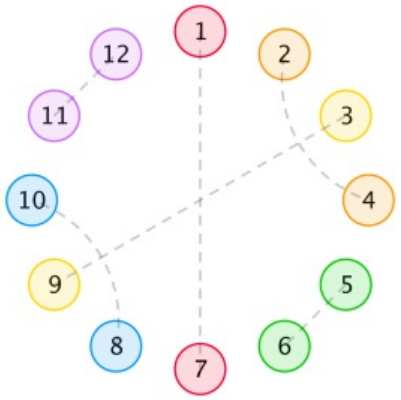
E. Days of Floral Colours

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

The Floral Clock has been standing by the side of Mirror Lake for years. Though unable to keep time, it reminds people of the passage of time and the good old days.

On the rim of the Floral Clock are $2n$ flowers, numbered from 1 to $2n$ clockwise, each of which has a colour among all n possible ones. For each colour, there are exactly two flowers with it, the *distance* between which **either is less than or equal to 2, or equals n**. Additionally, if flowers u and v are of the same colour, then flowers opposite to u and opposite to v should be of the same colour as well — symmetry is beautiful!

Formally, the *distance* between two flowers is 1 plus the number of flowers on the minor arc (or semicircle) between them. Below is a possible arrangement with $n = 6$ that cover all possibilities.



The *beauty* of an arrangement is defined to be the product of the lengths of flower segments separated by all opposite flowers of the same colour. In other words, in order to compute the beauty, we remove from the circle all flowers that have the same colour as flowers opposite to them. Then, the beauty is the product of lengths of all remaining segments. Note that we include segments of length 0 in this product. If there are no flowers that have the same colour as flower opposite to them, the beauty equals 0. For instance, the *beauty* of the above arrangement equals $1 \times 3 \times 1 \times 3 = 9$ — the segments are $\{2\}$, $\{4, 5, 6\}$, $\{8\}$ and $\{10, 11, 12\}$.

While keeping the constraints satisfied, there may be lots of different arrangements. Find out the sum of *beauty* over all possible arrangements, modulo $998\,244\,353$. Two arrangements are considered different, if a pair (u, v) $(1 \le u, v \le 2n)$ exists such that flowers u and v are of the same colour in one of them, but not in the other.

Input

The first and only line of input contains a lonely positive integer n $(3 \le n \le 50\,000)$ — the number of colours present on the Floral Clock.

Output

Output one integer — the sum of *beauty* over all possible arrangements of flowers, modulo $998\,244\,353$.

Examples

input
3
output
24

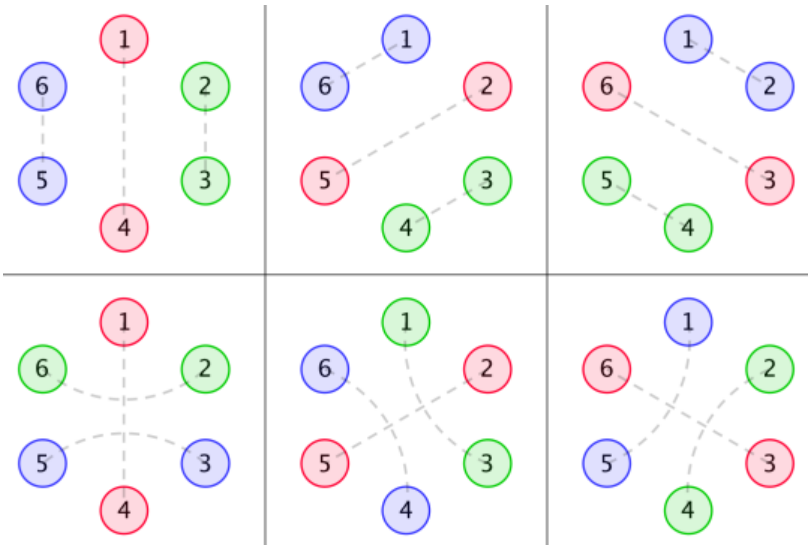
input
4
output
4

input
7
output
1316

input
15
output
3436404

Note

With $n = 3$, the following six arrangements each have a *beauty* of $2 \times 2 = 4$.



While many others, such as the left one in the figure below, have a *beauty* of 0. The right one is invalid, since it's asymmetric.