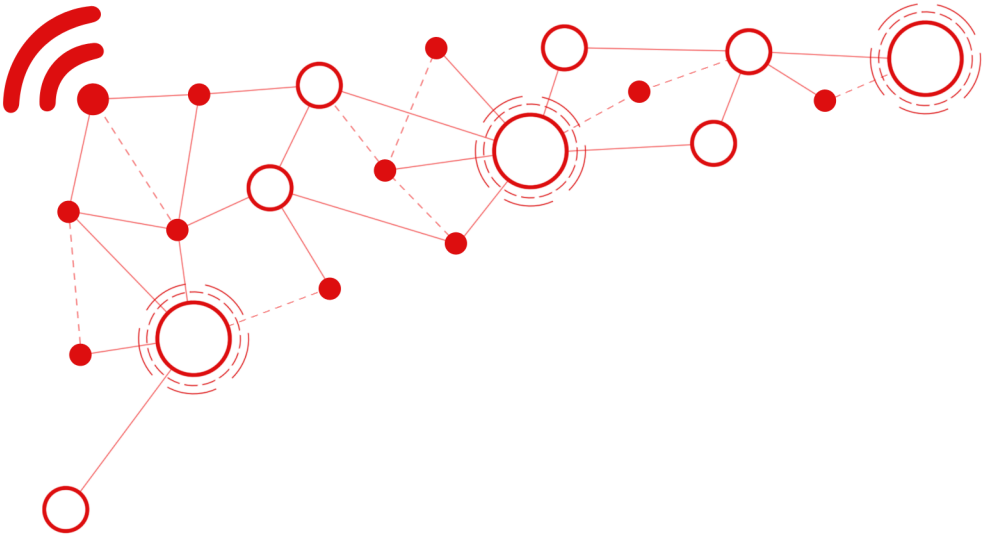
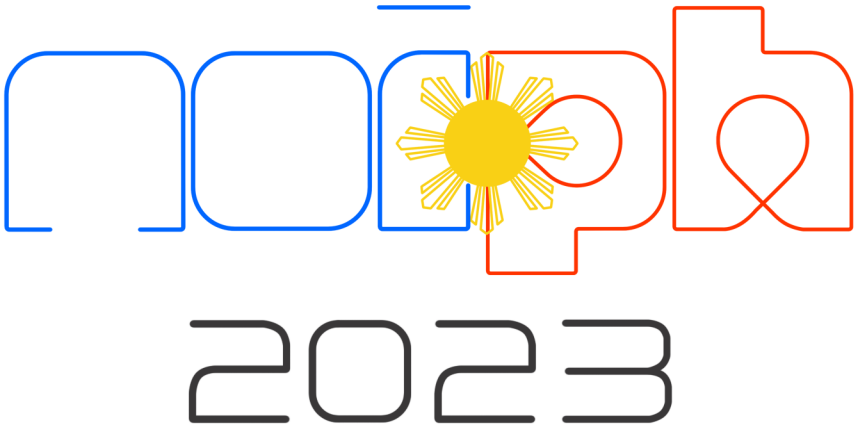
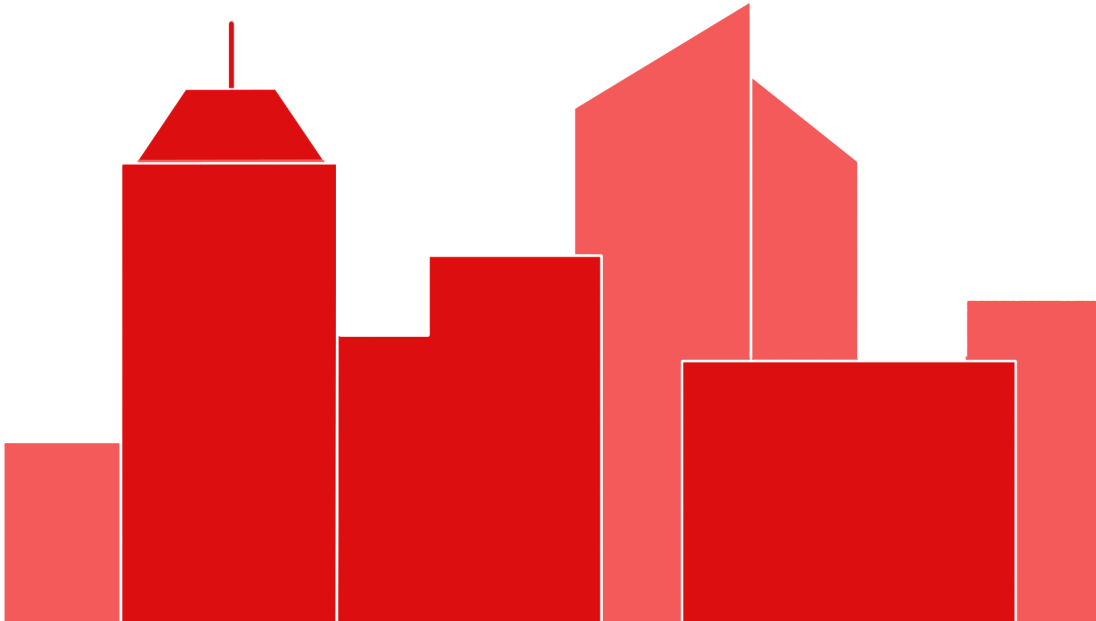




National Olympiad in Informatics

Revenge Round



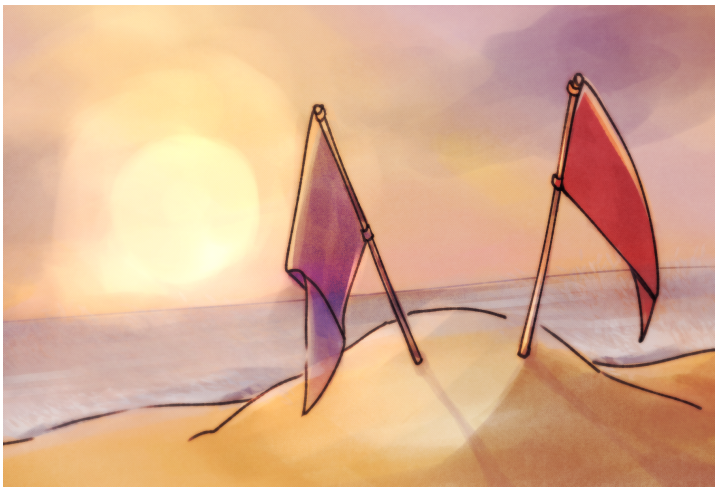
Contents

Notes

- Many problems have large input file sizes, so use fast I/O. In C++, use:
  - `ios_base::sync_with_stdio(false);` and
  - `cin.tie(nullptr);`
- On interactive problems, make sure to **flush** your output stream after printing. In C++, use `fflush(stdout);` or `cout << endl;`
- Make sure you have PyPy 3.9 in your system, and the command `pypy3.9` is working in the terminal.
- The judge is sometimes somewhat *strict*. Please terminate each line with the `'\n'` character, and please don't print trailing whitespace in any line.
- Good luck and enjoy the contest!

Problem A

Harry Palayok and the Sedimentary Stone



Harry Palayok has been living on the skirts of Manila Bay for the past 6 years. Although he lives a simple life, his memories of grade school have taught him the power of magic. He particularly believes that good mental health and well-being are some of the most powerful forms of magic there is.

He therefore wants to uplift the spirits of everyone around him—and what better way to do this than by beautifying his surroundings? Having received a shipment of dolomite from an anonymous donor, he begins his work.

The area of Manila Bay he plans to work on can be described as an h by w grid. At the end of the reclamation process, each cell will either be empty, or be filled with dolomite.

Because Harry wants to share his joy with everyone, he wants as many people as possible to visit the bay. He defines a *viewing site* to be a cell such that:

- The cell does not contain dolomite. Obviously, they'd need some place to stand on!
- None of the 4 cells *orthogonally* adjacent (i.e. up, down, left, right) have dolomite in them. This lets visitors walk in and out of the site freely.
- Exactly 3 of the 4 *diagonally* adjacent cells have dolomite in them. This lets visitors experience its healing effects, without being overloaded too much.

Note that cells that are on a border or corner of the grid cannot be viewing sites.

Since he wishes to improve the mental health of as many citizens as possible, Harry will lay dolomite on the Bay such that the number of viewing sites is maximized. Can you provide a layout of which cells to fill, to help him fulfill this?

Input Format

The only line of input contains two space-separated integers h and w: the height and width of the grid, respectively.

Output Format

Output h lines, each containing w characters, representing the optimal layout. The jth character of the ith line must be "`#`" if the cell at row i and column j contains dolomite, and "`.`" otherwise.

If there are multiple optimal layouts, you may output any of them.

Constraints and Subtasks

For all subtasks
$1 \leq w \leq 60$ $1 \leq h \leq 14$

Subtask	Points	Constraints
1	**4**	$h \leq 2$
2	**10**	$h \leq 5$, $w \leq 5$
3	**3**	$h \leq 3$
4	**3**	$h \leq 4$
5	**8**	$h \leq 5$
6	**8**	$h \leq 6$
7	**8**	$h \leq 7$
8	**8**	$h \leq 8$
9	**8**	$h \leq 9$
10	**8**	$h \leq 10$
11	**8**	$h \leq 11$
12	**8**	$h \leq 12$
13	**8**	$h \leq 13$
14	**8**	No additional constraints.

Sample I/O

Input 1	Output 1
3 3	#.# ... #..

Explanation

The cell at row 2 and column 2 is a viewing site, since all of its orthogonal neighbors have no dolomite, and only its bottom-right neighbor does not have dolomite. Since it can be shown that 1 is the maximum possible number of viewing sites, this output is considered correct.

Problem B
...Are the Bestagons!

*What is **objectively** the best shape?*

You have six tries, and the first five don't count.

You have somehow found yourself dragged into a hexagon-loving cult. The cult has n members, indexed from 1 to n (because who needs a name and an identity, when you have hexagons?). You are aware of m friendships within the cult; each friendship involves two people u and v, and says that u and v are friends with each other (this is a mutual relationship).

Hexagons are a source of immense power. A *hexagon* is a sequence of any six **different** people a, b, c, d, e, f, such that all of the following hold:

- a and b are friends
- b and c are friends
- c and d are friends
- d and e are friends
- e and f are friends
- f and a are friends

There may or may not be other friendships among these chosen six, but it doesn't matter. So long as *these* six friendships exist, then the six form a *hexagon*!

So tell me... how many hexagons does this cult have?

Formally, find the number of *sextuplets* (a, b, c, d, e, f) of different people that satisfy the conditions above. Since the answer can be large, only output it modulo 998244353.

(No, there's no subplot about escaping. I think you're just stuck here now.)

Input Format

The first line of input contains two space-separated integers n and m, the number of cult members and the number of friendships.

Then, m lines of input follow, each describing some friendship. Each line contains two space-separated integers u and v, specifying that u and v are friends with one another.

Output Format

Output a single integer denoting the number of hexagons the cult have, modulo 998244353.

Constraints and Subtasks

For all subtasks
$0 \le n \le 500$ $0 \le m \le n(n-1)/2$ $1 \le u, v \le n$ No friendship is mentioned more than once in the input. (This includes the fact that if `u v` appears, then `v u` will not.) A person is never described to be friends with themself.

Subtask	Points	Constraints
1	**14**	$n \le 20$
2	**17**	$n \le 52$
3	**19**	$m \le 500$
4	**12**	$n \le 120$
5	**38**	No additional constraints.

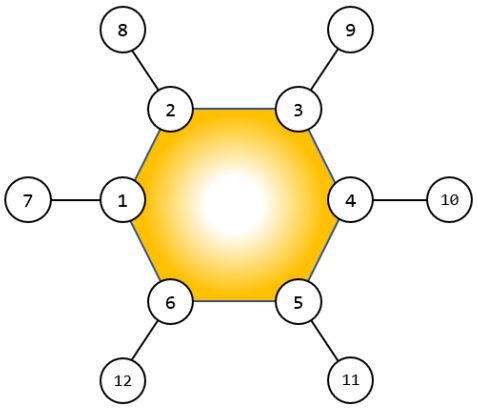
Sample I/O

Input 1	Output 1
12 12 1 2 2 3 3 4 4 5 5 6 1 6 7 1 8 2 9 3 10 4 11 5 12 6	12

Input 2	Output 2
6 6 1 2 2 3 3 4 4 5 5 1 1 6	0

Explanation

The following diagram showcases the first sample input. One hexagon has been highlighted: $a = 1$, $b = 2$, $c = 3$, $d = 4$, $e = 5$, and $f = 6$.



Problem C
Kelly Kawno and the Circuit Factory

Welcome to Kelly Kawno's Circuit Factory! Kelly Kawno is the mastermind behind the greatest circuit factory on Earth. Everywhere you look, it's different kinds of circuits, as far as the eye can see. Unfortunately, Ms Kelly is busy giving a tour right now to some other five children who found golden tickets in their graphics cards, so you'll have to settle for me as your guide today. Shall we begin?

Circuits are (more or less) just Boolean expressions. A Boolean expression is just like an arithmetic expression, except all values can only be either `1` or `0` (as such, we call these "Boolean values"). The circuits in our factory accept **at most** 10 variables $x_0, x_1, x_2, \ldots, x_9$ as input. Then, we can perform *operations* on these values. The `AND`, `OR`, `NAND`, and `NOR` operators each accept two Boolean values as input, and produce one Boolean value as output. Each one's behavior is summed up in the following tables, where `a` and `b` are the input values.

a	b	a b AND
0	0	0
0	1	0
1	0	0
1	1	1

a	b	a b OR
0	0	0
0	1	1
1	0	1
1	1	1

a	b	a b NAND
0	0	1
0	1	1
1	0	1
1	1	0

a	b	a b NOR
0	0	1
0	1	0
1	0	0
1	1	0

The resulting value after evaluating some operator can be further used as input for *other* operators. This nesting allows us to create some pretty intricate expressions! For example, consider the following Boolean expression.

```
x0 x1 AND x2 x1 x2 NOR OR NAND x0 AND
```

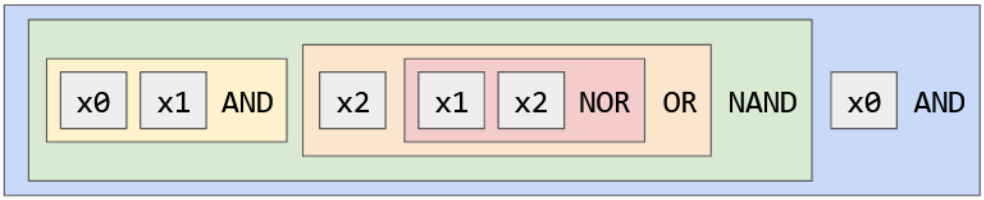
Note our use of *postfix* notation here, where we place the operator *after* its two arguments. This is in contrast to the *infix* notation that you might be more familiar with from arithmetic, where the operator goes *between* its two operands. Postfix notation has a lot of advantages over infix notation:

- Implementing a parser for it is significantly easier. Try it!
- Even without parentheses or an explicit PEMDAS-like convention, the order of operations is always completely unambiguous.

If we were to place explicit parentheses in the above expression, then it would be:

```
(((x0 x1 AND) (x2 (x1 x2 NOR) OR) NAND) x0 AND)
```

However, parentheses are not needed. The following diagram shows how to break down our example Boolean expression. See how the expressions' nestedness can be implied by the notation, even without explicit parentheses.



As an exercise, you can verify that if $x_0 = 1$, $x_1 = 0$, and $x_2 = 1$, then the above Boolean expression evaluates to 1.

We have many different kinds of circuits in this factory. But practically speaking, there is a sense in which two different circuits might be considered "the same". We say that two circuits are *equivalent* if they both produce the *same* output for any configuration of inputs. For example, you can verify that our example Boolean expression from earlier is equivalent to the following, simpler expression:

```
x0 x1 x2 NAND AND
```

In our factory, we partition all of our circuits into different *production lines*, where two circuits belong to the same production line if and only if they are equivalent.

Oh, Ms. Kawno is back! It seems that she was looking for someone to inherit her Circuit Factory, but all five children had failed her challenge. Perhaps you would like to try?

Ms. Kawno has selected twenty different circuits. Your challenge is to create a circuit that is equivalent to each one, using **as few** operators as possible.

Oh, and also, to make it more exciting, Ms. Kawno has required that you **only use** her new experimental operators, which she calls `BLAND` and `BLOR`. Their behavior is summed up in the following tables—be careful, the operators aren't commutative!

a	b	a b BLAND
0	0	0
0	1	1
1	0	0
1	1	0

a	b	a b BLOR
0	0	1
0	1	1
1	0	0
1	1	1

Why are they called `BLAND` and `BLOR`? Consider that an extra puzzle from Ms. Kawno, free of charge!

In summary, for each of the given circuits, construct an equivalent one that **only** uses `BLAND` and `BLOR`, and which does so using as few operators as possible. Specifically, Ms. Kawno herself was able to replicate each circuit, and her operator count will serve as a "target" which you need to try to match (or surpass!).

Input and Output Format

This is an output-only problem. You will be provided with twenty test files, each containing a test case. Each test file follows the file name format of `input_XX.txt`, and for each, you submit to it by uploading a file `output_XX.txt`, containing an equivalent `BLAND/BLOR` circuit. You can get the input files by downloading and extracting `circuit_inputs.zip` from CMS.

Each test file and each submission should consist of a single line, containing a valid Boolean expression in postfix notation. To be absolutely explicit, we define valid Boolean expressions in postfix notation as follows:

- `xi` is a valid expression, where `i` is a digit from `0` to `9`. This represents the value of the variable x_i.
- If `a` and `b` are valid expressions, then `a b op` is also a valid expression, whose value is the result of evaluating the operator `op` with `a` as the first argument, and `b` as the second argument.
  - For test files, `op` is guaranteed to be one of `AND`, `OR`, `NAND`, `NOR`.
  - For your submissions, `op` must only be either `BLAND` or `BLOR`.

Each test file contains no more than 10^5 operators.

Your submissions should each contain no more than 10^5 operators.

Summaries of the twenty test cases are provided below.

- **Tests 1 to 4** contain exactly one operator.
- **Tests 5 to 8** contain only one *kind* of operator
- For the remaining tests, consider the production lines whose output values only depend on $x_0, x_1, \ldots, x_{k-1}$. Among them, four production lines were uniformly randomly sampled, and from each, we selected a circuit which only uses the variables $x_0, x_1, \ldots, x_{k-1}$.
  - **Tests 9 to 12** have $k = 2$
  - **Tests 13 to 16** have $k = 6$
  - **Tests 17 to 20** have $k = 10$

Scoring

Each correct test case is worth up to 5 points. Each test case has a "target value" T, meaning you should be aiming to use T or fewer operators in your submission.

If your submission to a test case is correct, then you are awarded a baseline of 1 point (yay!). Let c be the number of operators used in your submission. The remaining 4 points are then awarded based on how close you managed to bring c

to T. In general, a smaller c is better, with full points being awarded when $c \leq T$. The precise scoring formula is given below.

$$\begin{cases} 0, & 10^5 < c, \\ 1, & 70T < c \leq 10^5, \\ 1 + 4\left(\dfrac{70T - c}{70T - T}\right)^7, & T < c \leq 70T, \\ 5, & c \leq T. \end{cases}$$

Your score is rounded *down* to 3 decimal places.

If your submission is incorrect, then you will get 0 points for that test file.

The different targets T per test case are summarized in the following table.

Test Cases	T
1 to 4	2
5 to 6	10
7	475
8	350
9 to 12	2
13	38
14	39
15	34
16	39
17	558
18	562
19	550
20	559

Sample I/O

Input

```
x1 x0 NOR x2 AND x1 OR
```

Output

```
x0 x1 BLAND x2 x0 BLOR BLAND x1 BLOR
```

Problem D
The Fibonacci f-Tuplets

The problem statement for the original version of this problem was about the Pisano sisters from the adaptation of the manga of the same name, "The Fibonacci Five-tuplets". This is unacceptable. Kids these days watch too much anime! Tonight, we're going to watch a classic Filipino comedy movie, later a film series—The Montecillo ***Dodecatuplets****.*

Ina Montecillo has twelve children,

- *Juan Agustin*
- *Gertrudis "Tudis" Valdez*
- *Dimitri "Tri" Aquino*
- *Portia "Por" Evangelista*
- *Tirso "Pip" Uytingco*
- *Sixto "Six" Acueza*
- *Severina "Seven" Magdayao*
- *Catherine "Cate" Dalrymple*
- *Samuel "Shammy" Manio*
- *Martin "Ten-Ten" Kadooka*
- *Connie Calma*
- *Sweet Calma*

(Don't ask why none of them have the last name Montecillo.)

Ina's birthday is coming up soon, and f of her 12 children agreed that they would each get her a Fibonacci number as a gift! The Fibonacci numbers F_n are the elements of the sequence with $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$.

A shop near Quiapo has n different necklaces for sale, indexed from 1 to n, and each one containing a Fibonacci number. Given an array $a_1, a_2, a_3, \ldots, a_n$, the number on the ith necklace is F_{a_i}. This array a may contain duplicates.

Of these f siblings, all but one (so $f - 1$ of them) agree on two integers L and R. They promise each other to only buy necklaces whose indices lie between L and R inclusive, and they also will respect their birth order with the indices of the necklaces they buy. Formally, they can choose any $f - 1$ indices $i_1, i_2, \ldots, i_{f-1}$ such that $L \leq i_1 < i_2 < \cdots < i_{f-1} \leq R$, and they will purchase the necklaces with those indices.

The remaining sibling, Juan, realizes that this is his chance to stand out from his siblings. He arranges to acquire—completely separately—a necklace that contains the number F_x. If the value of his necklace is greater than or equal to the values of *all* his siblings' necklaces combined, then surely he will be recognized as the favored son!

But Juan wonders whether this is even something that is likely to happen, or if it's just a fantasy. Specifically and formally, he needs you to help him answer the following question: Given the values of L, R, and x, **count** the number of $(f-1)$-tuples $i_1, i_2, \ldots, i_{f-1}$ such that both of the following hold:

- $L \leq i_1 < i_2 < \cdots < i_{f-1} \leq R$
- $\sum_{t=1}^{f-1} F_{a_{i_t}} \leq F_x$

Also, none of the plans are really set in stone, so you'll need to answer q different queries, each with their own L, R, and x. Good luck!

We remind you that f is given, but that we're counting the ways to choose $f - 1$ indices.

Also, each answer can be quite large, so find the answers modulo 998244353.

Input Format

The first line of input contains three space-separated integers f, n and q.

The second line of input contains n space-separated integers $a_1, a_2, a_3, \ldots, a_n$, describing which entries from the Fibonacci sequence are written on each necklace.

Then, q lines follow, each describing a query. Each line contains the three space-separated integers L, R, and x for that query.

Output Format

Output q lines, answering each of the queries in order.

Constraints and Subtasks

For all subtasks
$5 \leq n \leq 400000$ $1 \leq q \leq 160000$ $5 \leq f \leq 12$ $0 \leq x, a_i \leq 10^8$

Subtask	Points	Constraints
1	**4**	$f = 5$ $n, q \leq 100$ $x, a_i \leq 40$
2	**9**	$f \leq 6$ $n, q \leq 100$ $x, a_i \leq 40$
3	**9**	$f \leq 6$ $n, q \leq 100$
4	**9**	$f \leq 6$ $x, a_i \leq 40$
5	**9**	$f \leq 6$ In all queries, $L = 1$ and $R = n$
6	**12**	$f \leq 6$
7	**9**	$n, q \leq 600$ $x, a_i \leq 100$
8	**13**	$f \leq 8$
9	**13**	$f \leq 10$
10	**13**	No additional constraints.

Sample I/O

Input 1	Output 1
5 5 2 1 3 5 2 7 1 5 20 1 5 2	5 0

Input 2	Output 2
5 8 3 5 6 7 8 8 7 6 5 1 8 10 2 7 10 1 1 10	60 9 0

Input 3	Output 3
6 8 3 5 6 7 8 8 7 6 5 1 8 10 2 7 10 1 1 10	24 0 0

Explanation

For the first sample input, with $a = \{1, 3, 5, 2, 7\}$, the actual values of the necklaces are $\{1, 2, 5, 1, 13\}$.

For the queries:

1. We have $L = 1$ and $R = 5$, and Juan's necklace has value $F_{20} = 6765$
2. We have $L = 1$ and $R = 5$, and Juan's necklace has value $F_2 = 1$