The N Lives of Raichu

This is a Communication problem

Raichu is a very friendly and affectionate cat who has lived many lives. But the end has come, and you must help Raichu determine which is the last of its lives.

Problem

The jury has a binary array a of size M, initially all values are 0. You must implement the function *Raichu()*. Your program will be executed N times, and each time, the jury will call the function *Raichu()* with the parameters N and M. Your program can call the functions *modificar()* and *leer()*. With these functions, you can modify the value of one of the positions of array a, or read a position of the array a. The function *Raichu()* returns a boolean and must return true only on the N-th execution of the program. The only information you have is the array a. Help Raichu determine which is its N-th and last life.

Implementation Details

You must implement the function *Raichu()*. This function receives two integers, N and M, and returns a boolean. Additionally, you can call the function *leer()*, which receives an integer $0 \leq i \leq M-1$ and returns the current value of $a[i]$, and the function *modificar()* which receives an integer $0 \leq i \leq M-1$ and an integer $0 \leq x \leq 1$, and makes $a[i] = x$. To carry out the interaction, you must include the library *"Raichu.h"* with the command *#include "Raichu.h"*.
An example of how the program would look is as follows:

```
#include "Raichu.h"
#include <bits/stdc++.h>
using namespace std;

bool Raichu(int N, int M) {
    // Implement this function.
}
```

Evaluation Criteria

Each test case consists of N, M and ten fixed evaluation parameters $a_1, a_2, \cdots, a_{10}$.
If your program does not return true exactly on the last execution, and false on all other executions, you will receive 0 points.
For each test case, we define C_1 as the maximum number of different positions in the binary array that you involve in each time your program is executed.

We say that during an execution you involved position i, if you called the function *modificar()* or *leer()* with position i (if you call both it still counts as only one position). We also define C_2, as the maximum number of different positions in the binary array that you involve each time your program is executed, not counting the first execution. If your program effectively solves the problem, you will receive points based on:

- In the first test case, you will receive one point for each evaluation parameter a_i that satisfies $C_1 \leq a_i$.
- In the second test case, you will receive 1 point for each evaluation parameter a_i that satisfies $C_1 \leq a_i$, and 1 point for each one that satisfies $C_2 \leq a_i$.
- In the third test case, you will receive 2 points for each evaluation parameter a_i that satisfies $C_1 \leq a_i$, and 1 point for each one that satisfies $C_2 \leq a_i$.
- In the fourth test case, you will receive 2 points for each evaluation parameter a_i that satisfies $C_1 \leq a_i$, and 2 points for each one that satisfies $C_2 \leq a_i$.

In the following table, the parameters and specific considerations for each test case are listed:

Test Case	**Points**	N	M	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	**Time**
1	10	2^{10}	10	10	10	10	10	10	10	10	10	10	10	3s
2	20	2^{16}	10^5	14	13	12	11		9	8	7	6	6	4s
3	30	2^{20}												5s
4	40	2^{26}												14s