Las N vidas de Raichu

Este es un problema de Comunicación

Raichu es un gato muy simpático y cariñoso, que ha vivido muchas vidas. Pero ha llegado el final, y debes ayudar a Raichu a determinar cuál es la última de sus vidas.

Problema

El jurado tiene un arreglo binario a de tamaño M, inicialmente todos los valores son 0. Debes implementar la función *Raichu()* . Tu programa será ejecutado N veces, y cada vez, el jurado llamará la función *Raichu()* con los parámetros N y M. Tu programa puede llamar las funciones *modificar()* y *leer()*. con dichas funciones, puedes modificar el valor de una de las posiciones de a, o leer una posición del arreglo a. La función *Raichu()* regresa un booleano, y debe regresar verdadero únicamente en la N-ésima ejecución del programa. La única información que tienes es el arreglo a. Ayuda a Raichu a determinar cuál es su N-ésima y última vida.

Detalles de Implementación

Debes implementar la función *Raichu()*. Esta función recibe dos enteros, N y M, y regresa un booleano. Además, puedes llamar la función *leer()*, que recibe un entero $0 \leq ileM - 1$ y regresa el valor actual de $a[i]$, y la función *modificar()* que recibe un entero $0 \leq i \leq M - 1$ y un entero $0 \leq x \leq 1$, y hace que $a[i] = x$. Para llevar a cabo la interacción, debes agregar la librería *"Raichu.h"* con el comando *#include "Raichu.h"*. Un ejemplo de cómo se vería el programa es el siguiente:

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

bool Raichu(int N, int M) {
    // Implementa esta función.
}
```

Criterios de Evaluación

Cada caso de prueba consiste de N, M y diez parámetros de evaluación fijos $a_1, a_2, \cdots, a_{10}$. Si tu programa no regresa verdadero exactamente en la última ejecución, y falso en todas las demás, obtendrás 0 puntos.
Durante un caso de prueba, definimos C_1 como la máxima cantidad de posiciones distintas del arreglo binario que involucras en cada una de las veces que se ejecuta tu programa.

Decimos que durante una ejecución involucraste la posición i, si llamaste la función *modificar()* o *leer()* con la posición i (si llamas las dos sigue contando como solamente una posición).
Definamos también, C_2, como la máxima cantidad de posiciones distintas del arreglo binario que involucras a partir de la segunda vez que se ejecuta tu programa.
Si tu programa resuelve efectivamente el problema, obtendrás puntos basado en:

- En el primer caso de prueba, obtendrás un punto por cada parámetro de evaluación a_i que cumpla con $C_1 \leq a_i$.
- En el segundo caso de prueba, obtendrás 1 punto por cada parámetro de evaluación a_i que cumpla con $C_1 \leq a_i$, y 1 punto por cada uno que cumpla $C_2 \leq a_i$.
- En el tercer caso de prueba, obtendrás 2 puntos por cada parámetro de evaluación a_i que cumpla con $C_1 \leq a_i$, y 1 punto por cada uno que cumpla $C_2 \leq a_i$.
- En el cuarto caso de prueba, obtendrás 2 puntos por cada parámetro de evaluación a_i que cumpla con $C_1 \leq a_i$, y 2 puntos por cada uno que cumpla $C_2 \leq a_i$.

En la siguiente tabla, están los parámetros y consideraciones específicas para cada caso de prueba:

caso	**puntos**	N	M	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	**tiempo**
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