

Juego en el aula

Un prestigioso instituto de Tashkent celebra su aniversario organizando un partido entre alumnos y profesores. Hay N alumnos sentados en el aula, numerados del 0 al $N - 1$. Cada alumno tiene en la mano una hoja de papel que contiene un arreglo de números enteros. Inicialmente, cada hoja está en blanco, por lo que contiene un arreglo vacío.

Los alumnos juegan con M profesores, numerados del 0 al $M - 1$, y con el director. El juego se desarrolla en M pasos, también numerados del 0 al $M - 1$. En el paso j , el profesor j entra en el aula y tienen lugar los siguientes acontecimientos:

- Algunos alumnos (posiblemente ninguno) levantan la mano. Se garantiza que en cada paso al menos un alumno **no** levante la mano, y cada alumno puede levantar la mano **como máximo una vez** durante todo el juego.
- El profesor examina los papeles que los alumnos tienen en ese momento y puede **modificar** los papeles de los alumnos que **no** han levantado la mano en este paso. Para cada uno de esos alumnos, el profesor puede sustituir el contenido de su papel por un nuevo arreglo (posiblemente vacío) de números enteros. Cada número entero debe estar comprendido entre 0 y 63 , inclusive, y el arreglo puede tener como máximo 63 elementos.
- Tras estas modificaciones, el profesor j se retira y los alumnos intercambian sus hojas según una permutación secreta P . Es decir, cada alumno i ($0 \leq i < N$) entrega su hoja al alumno $P[i]$, donde $P[0], P[1], \dots, P[N - 1]$ son N números enteros **distintos** comprendidos entre 0 y $N - 1$, inclusive. Los valores de P son **fijos** durante todas las fases del juego, pero ni los profesores ni el director los conocen.

Una vez finalizadas todas las M etapas y realizado el último intercambio según P , entra el director. Mirando únicamente los papeles que tienen los alumnos, el director debe determinar, para cada alumno i ($0 \leq i < N$), el número de etapa j en el que el alumno i levantó la mano, o concluir que el alumno i nunca levantó la mano.

Los profesores no pueden comunicarse entre sí ni con el director, salvo a través de los arreglos de números enteros escritos en los papeles. Cada profesor conoce el paso en el que entra en el aula.

Tu tarea consiste en diseñar e implementar una estrategia para los profesores y el director que identifique correctamente si cada alumno levantó la mano y en qué momento lo hizo. Tu puntuación dependerá de la longitud máxima de cualquier arreglo escrito en una hoja: una longitud máxima más corta da lugar a una puntuación mayor o igual.

Detalles de la implementación

Debes implementar dos procedimientos, uno para los profesores y otro para el director.

El procedimiento que debes implementar para los **profesores** es:

```
std::vector<std::vector<int>> process_step(  
    int N, int M, int R,  
    std::vector<int> T,  
    std::vector<std::vector<int>> A)
```

- N : el número de alumnos.
- M : el número de pasos, que es también el número de profesores.
- R : el número de paso actual (desde 0 hasta $M - 1$).
- T : un arreglo (posiblemente vacío) formado por los índices de los alumnos que levantan la mano en este paso, ordenados de menor a mayor.
- A : un arreglo de longitud N que describe los papeles, donde $A[i]$ ($0 \leq i < N$) es el arreglo de números enteros que figura en el papel que tiene el alumno i al comienzo del paso.
- Este procedimiento se invoca exactamente M veces por partida, en el orden $R = 0, 1, \dots, M - 1$.

El procedimiento debe devolver un arreglo B de longitud N , que represente los trabajos tras las modificaciones del profesor, en el mismo formato que A .

- Para cada alumno i que haya levantado la mano (es decir, i aparece en T), el trabajo no debe modificarse, por lo que $B[i] = A[i]$.
- Para cada i tal que $0 \leq i < N$, la longitud de $B[i]$ no debe superar 63, y cada número entero de $B[i]$ debe estar comprendido entre 0 y 63, inclusive.

El procedimiento que debes implementar para el **director** es:

```
std::vector<int> determine_steps(  
    int N, int M, std::vector<std::vector<int>> A)
```

- N, M : igual que arriba.
- A : un arreglo de longitud N , donde $A[i]$ es el arreglo de números enteros que figura en el papel que tiene el alumno i tras todos los pasos M .
- Este procedimiento se llama exactamente una vez por partida, tras la última llamada a `process_step`.

El procedimiento debe devolver un arreglo D de longitud N . Para cada i tal que $0 \leq i < N$, debe cumplirse que

- $D[i] = j$ si el alumno i levantó la mano durante el paso j , o

- $D[i] = -1$ si el alumno i nunca levantó la mano durante toda la partida.

Tu programa no puede almacenar ni transmitir ninguna información de una llamada a `process_step` a otra, salvo a través del valor de retorno de `process_step`. Si tu programa intenta hacerlo, tu puntuación en CMS podría ser incorrecta y podría reducirse una vez finalizado el concurso. Ten en cuenta que el sistema de corrección puede ejecutar varias instancias de tu programa simultáneamente. Esto significa que las llamadas a `process_step` y `determine_steps` de la misma partida pueden ejecutarse en instancias diferentes, y que las llamadas a `process_step` y `determine_steps` de partidas diferentes pueden ejecutarse en la misma instancia. Cada caso de prueba incluye hasta 5 partidas.

Restricciones

- $2 \leq N \leq 63$
- $1 \leq M \leq 63$
- Cada alumno levanta la mano **como máximo una vez** durante toda la partida. En otras palabras, para cada alumno i , hay como máximo un paso j en el que i levanta la mano.
- En cada paso, al menos un alumno **no** levanta la mano.

Subtareas y puntuación

Subtarea	Puntuación	Restricciones adicionales
1	4	$M = 1$
2	6	$N = 2$
3	9	$P[i] = i$ para todos los $0 \leq i \leq N - 1$
4	25	Como máximo, un estudiante levanta la mano en cada paso.
5	56	Sin restricciones adicionales.

Para cada caso de prueba, tu puntuación será 0 (que en CMS aparece como `Output isn't correct`) si el valor de retorno de cualquier llamada al procedimiento `process_step` no es válido o si el valor de retorno de cualquier llamada al procedimiento `determine_steps` es incorrecto.

En caso contrario, sea C la longitud máxima de **cualquier** arreglo en **cualquier** B devuelta por `process_step`. Entonces, la puntuación de un caso de prueba en una subtarea con puntuación S se calcula como $S \cdot X$, donde X depende de C según la siguiente tabla:

Condición	X
$C \leq 2$	1.00
$C = 3$	0.75
$C = 4$	0.55
$5 \leq C \leq 13$	$0.50 - 0.03 \cdot (C - 5)$
$14 \leq C \leq 63$	$0.19 \cdot \frac{64-C}{64-14} + 0.04$

Ejemplo

Consideremos un escenario con $N = 4$ alumnos, $M = 2$ profesores y la permutación $P = [0, 3, 1, 2]$. Inicialmente, todos los exámenes están en blanco.

El grader llama primero a:

```
process_step(4, 2, 0, [1], [[], [], [], []])
```

El alumno 1 ha levantado la mano, por lo que su examen no puede modificarse. El profesor 0 puede decidir escribir $[10]$ en el examen del alumno 0, escribir $[1, 63, 4]$ en el examen del alumno 3 y dejar en blanco el examen del alumno 2. Para ello, el procedimiento debe devolver $B = [[10], [], [], [1, 63, 4]]$.

Después de que el profesor 0 se marche, los alumnos intercambian sus exámenes según P . Tras el intercambio, los exámenes quedan así: $A = [[10], [], [1, 63, 4], []]$.

El grader ahora llama a:

```
process_step(4, 2, 1, [0, 3], [[10], [], [1, 63, 4], []])
```

Los alumnos 0 y 3 han levantado la mano, por lo que sus exámenes no pueden modificarse. El profesor 1 puede decidir escribir $[0, 40]$ en el examen del alumno 1 y escribir $[1, 50]$ en el examen del alumno 2. Para ello, la función debe devolver $B = [[10], [0, 40], [1, 50], []]$.

Después de que el profesor 1 se marche, los exámenes se vuelven a intercambiar según P . Tras el intercambio, los exámenes quedan en $A = [[10], [1, 50], [], [0, 40]]$.

Por último, el grader llama a:

```
determine_steps(4, 2, [[10], [1, 50], [], [0, 40]])
```

Este procedimiento debe devolver el arreglo $D = [1, 0, -1, 1]$, ya que los alumnos 0 y 3 levantaron la mano en el paso 1, el alumno 1 levantó la mano en el paso 0 y el alumno 2 nunca levantó la mano. En este ejemplo, $C = 3$.

Grader de Ejemplo

Formato de entrada:

```
N M
Q[0] Q[1] ... Q[N-1]
P[0] P[1] ... P[N-1]
```

Aquí, Q es un arreglo de longitud N , donde $Q[i] = j$ si el estudiante i levanta la mano durante el paso j , o $Q[i] = -1$ si el estudiante i nunca levanta la mano durante toda la partida.

Formato de salida:

Tras cada llamada `aprocesar_step`, el grader de ejemplo muestra el contenido de los exámenes. Sea K la longitud de B y $L[i]$ la longitud de $B[i]$ (para cada $0 \leq i < K$). El grader de ejemplo imprime B en el siguiente formato, **seguido de una línea en blanco**:

```
K
L[0] B[0][0] B[0][1] ... B[0][L[0]-1]
L[1] B[1][0] B[1][1] ... B[1][L[1]-1]
:
L[K-1] B[K-1][0] B[K-1][1] ... B[K-1][L[K-1]-1]
```

A continuación, el grader de ejemplo intercambia los trabajos según la permutación P . Si $K \neq N$, el grader de ejemplo imprime un mensaje de error y finaliza la ejecución.

Tras llamar a `determine_steps`, el grader de ejemplo muestra:

```
C H
D[0] D[1] ... D[H-1]
```

Aquí, H es la longitud del arreglo D devuelta por `determine_steps`.