

교실 게임

타슈켄트의 명문 고등학교 하나가 개교기념일을 맞아 학생과 교사가 같이 하는 게임을 한다. 학생은 모두 N 명이 교실에 있고, 0부터 $N - 1$ 까지 번호가 매겨져 있다. 각각의 학생은 정수의 배열이 쓰여진 종이를 갖고 있다. 처음에는, 모든 종이는 백지상태이다. 즉, 빈 배열을 저장하고 있다.

학생은 교사 M 명과 같이 게임을 한다. 교사는 0부터 $M - 1$ 로 번호가 매겨져 있다. 그리고 교장 한 명이 있다. 이 게임은 M 단계로 이루어진다. 단계도 0부터 $M - 1$ 로 번호가 매겨진다. 단계 j 에서, 교사 j 가 교실에 들어오면 다음과 같은 일이 일어난다.

- 0명 이상 $N - 1$ 명 이하의 학생이 손을 든다. 다시 말하면, 매 단계마다 최소한 한 명은 손을 **들지 않는다**. 게임이 끝날 때까지 각각의 학생은 **최대 한 번** 손을 들 수 있다.
- 교사 j 는 학생들이 갖고 있는 종이를 보고, 손을 **들지 않은** 학생들이 갖고 있는 종이를 **고칠** 수 있다. 각 학생에 대해서 교사는 종이에 써진 내용을 새로운 배열로 (비어있는 배열일 수도 있다) 바꿀 수 있다. 쓰는 정수는 0 이상 63 이하이어야 하며, 배열은 최대 63개의 원소를 가질 수 있다.
- 이 과정을 마치고 교사 j 가 교실을 나간다. 다음에 학생들은 비밀 순열 P 에 의해서 종이를 교환한다. 즉, 각각 학생 i ($0 \leq i < N$)는 자신의 종이를 학생 $P[i]$ 에게 주는데, $P[0], P[1], \dots, P[N - 1]$ 는 N 개의 서로 다른 0 이상 $N - 1$ 이하인 정수이다. P 의 값은 게임을 하는 동안 **고정되어** 있지만 교사와 교장은 이 배열을 모른다.

M 단계가 모두 끝나고 교장이 들어온다. 교장은 학생들이 갖고 있는 종지만 보고서 각각의 학생 i ($0 \leq i < N$)가 손을 든 단계 j 를 알아내거나, 이 학생이 손을 든 적이 없다는 것을 알아내야 한다.

학생들이 들고 있는 종이에 쓰여진 정수의 배열 말고는 교사는 다른 교사 또는 교장과 소통할 방법이 없다. 모든 교사는 자기가 몇 번 단계에 들어가는지 알고 있다.

당신이 할 일은 교사들과 교장이 협동하여 정확히 언제 학생이 손을 들었는지 알 수 있는 전략을 고안해내는 것이다. 당신의 점수는 종이에 쓴 배열의 길이에 의해 결정되며, 길이가 짧을 수록 점수가 높아진다.

Implementation Details

다음 두 함수를 구현해야 한다. 하나는 교사에 대한 것이고 다른 하나는 교장에 대한 것이다.

교사에 대해 구현해야 하는 함수는 다음과 같다.

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

- N : 학생의 수.
- M : 단계의 수, 즉 교사의 수.
- R : 현재 단계 (0부터 $M - 1$ 까지).
- T : 이 단계에서 손을 든 학생들의 번호가 오름차순으로 정렬되어 있는 배열 (비어있는 배열일 수 있다).
- A : 종이에 쓰여진 정보를 기술하는 길이 N 인 배열로 $A[i]$ ($0 \leq i < N$)는 학생 i 가 이 단계가 시작할 때 들고 있는 종이에 쓰여진 배열이다.
- 이 함수는 한 게임에서 $R = 0, 1, \dots, M - 1$ 순서로 정확히 M 번 호출된다.

이 함수는 길이 N 인 배열 B 를 리턴해야 하는데, 이 배열은 교사가 고친 다음 종이들의 상태를 A 와 같은 포맷으로 표현한다.

- 손을 든 모든 학생 i 에 대해서 (즉, i 가 T 에 등장한다면), 종이의 내용은 바뀌어야 하지 않아야 하므로 $B[i] = A[i]$ 여야 한다.
- $0 \leq i < N$ 인 각각의 i 에 대해서, $B[i]$ 의 길이는 63을 넘어서는 안되며, $B[i]$ 에 들어있는 모든 정수는 0 이상 63 이하여야 한다.

교장에 대해 구현해야 하는 함수는 다음과 같다.

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

- N, M : 위와 같음.
- A : 길이 N 인 배열로, $A[i]$ 는 모든 M 단계가 끝났을 때 학생 i 가 들고 있는 종이에 쓰여진 배열이다.
- 이 함수는 한 게임에서 `process_step`가 마지막으로 호출된 다음 정확히 한 번 호출된다.

이 함수는 길이 N 인 배열 D 를 리턴해야 한다. $0 \leq i < N$ 인 각각의 i 에 대해서 다음이 성립해야 한다.

- 만약 학생 i 가 단계 j 에서 손을 들었다면 $D[i] = j$ 이고,
- 학생 i 가 손을 든 적이 없었다면 $D[i] = -1$ 이어야 한다.

당신의 프로그램은 `process_step`의 한 호출에서 다른 호출로 `process_step`의 리턴값 말고는 다른 정보를 전달할 수 없다. 만약 이를 시도한다면, CMS에서 당신의 점수는 정확하지 않을 것이며 대회가 끝난 뒤 감점될 수 있다. 그레이더가 당신의 프로그램 인스턴스 여럿을 동시에 실행시킬 수 있다는데 유의하라. 즉, 같은 게임에서 `process_step`, `determine_steps`의 호출이 다른 인스턴스에서 실행될 수 있고, 다른 게임에서 `process_step`, `determine_steps`의 호출이 같은 인스턴스에서 실행될 수 있다. 각 테스트케이스는 최대 5개의 게임으로 이루어진다.

Constraints

- $2 \leq N \leq 63$
- $1 \leq M \leq 63$
- 각각의 학생은 게임 전체에서 **최대 한 번** 손을 들 수 있다. 즉, 각각의 학생 i 는 자신이 손을 드는 단계 j 가 최대 한 개 존재한다.
- 각 단계에서, 최소한 한 명의 학생은 손을 **들지 않는다**.

Subtasks and Scoring

Subtask	Score	Additional Constraints
1	4	$M = 1$
2	6	$N = 2$
3	9	모든 $0 \leq i \leq N - 1$ 에 대해 $P[i] = i$.
4	25	각 단계마다 최대 한 명의 학생이 손을 든다.
5	56	추가적인 제약 조건이 없다.

각 테스트케이스에서, 만약 `process_step`나 `determine_steps`의 리턴값이 잘못되었다면 점수는 0점이며 CMS에서 `Output isn't correct`로 표시된다.

그렇지 않으면, C 가 `process_step`의 **모든** 호출에서 B 에 포함된 **모든** 배열의 길이의 최대값이라고 하자. 서브태스크에서 점수 S 인 테스트케이스 하나의 점수는 $S \cdot X$ 이며, X 는 C 값에 따라 다음과 같이 결정된다.

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

Example

학생 $N = 4$ 명, 교사 $M = 2$ 명, 순열 $P = [0, 3, 1, 2]$ 인 다음 시나리오를 생각해보자. 처음에는 모든 종이는 비어있다.

그레이더는 먼저 다음을 호출한다.

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

학생 1이 손을 들었기 때문에 이 학생의 종이는 고칠 수 없다. 교사 0은 학생 0의 종이에 [10], 학생 3의 종이에 [1,63,4]을 쓰고 학생 2의 종이는 빈 채로 남겨둔다. 이렇게 하려면 이 함수는 $B = [[10], [], [], [1,63,4]]$ 을 리턴해야 한다.

교사 0이 나간 다음, 학생들은 P 에 의해서 종이를 교환한다. 교환이 끝난 다음에 종이는 $A = [[10], [], [1,63,4], []]$ 이다.

그레이더는 이제 다음을 호출한다.

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

학생 0과 3이 손을 들었기 때문이 이들의 종이는 고칠 수 없다. 교사 1은 학생 1의 종이에 [0,40], 학생 2의 종이에 [1,50]을 쓴다. 이렇게 하려면 이 함수는 $B = [[10], [0,40], [1,50], []]$ 을 리턴해야 한다.

교사 1이 나간 다음, 학생들은 다시 P 에 의해 종이를 교환한다. 교환이 끝난 다음에 종이는 $A = [[10], [1,50], [], [0,40]]$ 이다.

마지막으로 그레이더는 다음을 호출한다.

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

이 함수는 배열 $D = [1, 0, -1, 1]$ 을 리턴해야 하는데 학생 0, 3이 단계 1에서, 학생 1이 단계 0에서 손을 들었고, 학생 2는 손을 들지 않았기 때문이다. 이 예제에서 $C = 3$ 이다.

Sample Grader

Input format:

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

Q 는 길이 N 인 배열로 만약 학생 i 가 단계 j 에서 손을 들었다면 $Q[i] = j$ 이고, 게임이 끝날 때까지 손을 든 적이 없다면 $Q[i] = -1$ 이다.

Output format:

각 `process_step` 호출이 끝나면, 샘플 그레이더는 종이의 내용을 출력한다. K 가 B 의 길이이고 $L[i]$ 가 $B[i]$ 의 길이라고 하자 (각 $0 \leq i < K$ 에 대해서). 샘플 그레이더는 B 를 다음 양식으로 출력하고 빈 줄 하나를 출력한다:

```
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]
```

샘플 그레이더는 다음에 P 에 의해서 종이를 교환한다. 만약 $K \neq N$ 이라면 샘플 그레이더는 에러를 출력하고 종료한다.

`determine_steps`를 호출한 다음 샘플 그레이더는 다음을 출력한다.

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

H 는 `determine_steps`가 리턴한 배열 D 의 길이이다.