

Scallion Pancake Party

Bohan is hosting a party. Since he loves scallion pancakes, he decides to buy them from the store nearby for his party. The store sells N different flavors of scallion pancakes, numbered from 0 to $N - 1$. Bohan decides to buy N scallion pancakes of each flavor, for a total of N^2 scallion pancakes. Each scallion pancake is divided into K identical slices before put into its own bag. The bags are numbered from 0 to $N^2 - 1$. Let $F[i]$ denote the scallion pancake flavor for bag i .

During the party, Bohan invites all guests to play a game. The game proceeds as follows.

First, Bohan picks N guests and brings them to rooms numbered $0, 1, \dots, N - 1$ such that each room has exactly one guest. The remaining guests stay outside until the end of the game. All N rooms look exactly the same, so the guests cannot tell which room they are in.

Next, for each room i ($0 \leq i < N$), Bohan enters room i and secretly tells the guest an integer $P[i]$, representing a **forbidden flavor**. He may or may not also tell the guest which room they are in. Bohan guarantees that $[P[0], P[1], \dots, P[N - 1]]$ is a permutation of $[0, 1, \dots, N - 1]$.

Then, there are N rounds. In the i -th round, Bohan brings bags $0, 1, \dots, N^2 - 1$ one by one in the sequential order into room $i - 1$. When the guest in room $i - 1$ receives bag j , they learn:

- $F[j]$, the scallion pancake flavor that bag j contains, and
- the number of slices remaining in the bag.

Before receiving the next bag, the guest in room $i - 1$ must decide how many slices to eat from the current bag. The number of slices they can eat depends on the situation:

- If $P[i - 1] \neq F[j]$, the guest can remove any number of slices from the bag and eat them.
- If $P[i - 1] = F[j]$, the guest is not allowed to eat any slices.

Note that the guests do not know the sequence $F[0], F[1], \dots, F[N^2 - 1]$ in advance.

After all N rounds, the guests outside are given the sequence of bags. Upon seeing the bags, they learn the scallion pancake flavors and the number of slices left in each bag. With this information, they must determine the values of $P[0], P[1], \dots, P[N - 1]$.

The guests are allowed to communicate before the game starts. They also know the values of N and K in advance. Your goal is to implement a strategy so that the guests outside can always correctly determine the values of $P[0], P[1], \dots, P[N - 1]$.

Implementation details

You need to implement three procedures.

You should implement the following two procedures for guests in rooms $0, 1, \dots, N - 1$.

```
void init(int N, int K, int p, int r)
```

Suppose the guest is in room i .

- N : the number of scallion pancake flavors.
- K : the number of slices of each scallion pancake before the game starts.
- $p = P[i]$: the forbidden flavor to eat in room i .
- If Bohan decides to tell the guest which room they're in, then $r = i$. Otherwise, $r = -1$.

```
int strategy(int b, int f, int s)
```

Suppose the guest is in room i .

- b : the current bag number.
- $f = F[b]$: the scallion pancake flavor for bag b .
- s : the number of slices left in bag b .
- This procedure should return a non-negative integer x , representing the number of slices the guest decides to eat.
 - If $f \neq P[i]$, then $0 \leq x \leq s$ must hold.
 - If $f = P[i]$, then $x = 0$ must hold.

You should implement the following procedure for the guests outside.

```
std::vector<int> guess(int N, int K, std::vector<int> F,  
                      std::vector<int> S)
```

- N : the number of scallion pancake flavors.
- K : the number of slices of each scallion pancake before the game starts.
- F : an array of length N^2 where $F[i]$ is the flavor of the scallion pancake in bag i .
- S : an array of length N^2 where $S[i]$ is the number of slices remaining in bag i after all N rounds.
- This procedure should return an array P of length N where $P[i]$ is the number given to the guest in room i .

Each test case involves T independent games.

During the judging process, for each of the T games, there will be $N + 1$ processes. For each i such that $0 \leq i < N$, process i represents the guest in room i . Process N represents the guests outside. For each i such that $0 \leq i < N$, process $(i + 1)$ starts after process i finishes.

For processes 0 to $N - 1$:

- `init` is called exactly once.
- `strategy` is called N^2 times after `init`.
 - It is guaranteed that for the j -th call, $b = j - 1$.

For process N :

- `guess` is called exactly once.

The judge may interleave processes from different games, but it is guaranteed that the relative order of processes within each individual game is preserved.

Constraints

- $1 \leq T \leq 400$.
- $2 \leq N \leq 30$.
- The sum of N^3 over all games in a test case doesn't exceed 27 000.
- $1 \leq K \leq N$
- $K \in \{1, 3, N\}$.
- $[P[0], P[1], \dots, P[N - 1]]$ is a permutation of $[0, 1, \dots, N - 1]$.
- $0 \leq F[j] < N$ for each j such that $0 \leq j < N^2$.
- i appears exactly N times in $[F[0], F[1], \dots, F[N^2 - 1]]$ for each i such that $0 \leq i < N$.
- The judge is **not adaptive**, that is, $[P[0], P[1], \dots, P[N - 1]]$ and $[F[0], F[1], \dots, F[N^2 - 1]]$ are fixed before the first call to `init` is made.

Subtasks

Subtask	Score	Additional Constraints
1	8	$K = 1, N = 2$.
2	7	$K = 1$ and Bohan decides to tell everyone which room they're in.
3	14	$K = 1$ and $[F[i \cdot N], F[i \cdot N + 1], \dots, F[i \cdot N + N - 1]]$ is a permutation of $[0, 1, \dots, N - 1]$ for each i such that $0 \leq i < N$.
4	18	$K = N$.
5	21	$K = 3, N \geq 3$.
6	32	$K = 1$.

Example

Consider the situation in which $T = 1$ and $N = 2, K = 2, P = [1, 0], F = [1, 0, 0, 1]$ for the only game in the test case.

Let $S[i]$ denote current number of slices remaining in bag i . Initially, $S = [2, 2, 2, 2]$.

Suppose the guests determine the following strategy before the game starts.

- For guests in rooms, if they are allowed to eat the current scallion pancake:
 - If the current scallion pancake is the first one they can eat, then they will eat all remaining slices.
 - Otherwise, they will only eat one slice.
- For the guests outside:
 - If $S = [0, 0, 1, 1]$, then they will guess that $P = [1, 0]$.
 - Otherwise, they will guess that $P = [0, 1]$.

The game proceeds as follows.

First, the judge starts process 0 representing the guest in room 0 and calls `init(2, 2, 1, -1)`.

After initializing, the judge makes the following calls:

Procedure call	Return value
<code>strategy(0, 1, 2)</code>	0
<code>strategy(1, 0, 2)</code>	2
<code>strategy(2, 0, 2)</code>	1
<code>strategy(3, 1, 2)</code>	0

The first and fourth calls must return 0 since $P[0] = F[0] = F[3] = 1$.

After this process ends (round 1 finishes), $S = [2, 0, 1, 2]$.

Next, the judge starts process 1 representing the guest in room 1 and calls `init(2, 2, 0, -1)`. After initializng, the judge makes the following calls:

Procedure call	Return value
<code>strategy(0, 1, 2)</code>	2
<code>strategy(1, 0, 0)</code>	0
<code>strategy(2, 0, 1)</code>	0
<code>strategy(3, 1, 2)</code>	1

Note that the second and third calls must return 0 since $P[1] = F[1] = F[2] = 0$.

After this process ends (round 2 finishes), $S = [0, 0, 1, 1]$.

Finally, the judge starts process 2 representing the guests outside. The judge makes the following call:

Procedure call	Return value
<code>guess(2, 2, [1, 0, 0, 1], [0, 0, 1, 1])</code>	<code>[1, 0]</code>

The guests outside guessed the correct permutation. Thus, the test case is considered correct.

Note that this approach does not always make it possible for the guests outside to guess the permutation correctly.

The attachment package for this task contains a different sample input in addition to this example.

Sample Grader

The sample grader only supports **one game per test case**.

Input format:

```
N K
P[0] P[1] ... P[N-1]
F[0] F[1] ... F[N*N-1]
reveal
```

- `reveal` is either 0 or 1.
 - If `reveal` is 0, the grader will act as if Bohan tells no one their room number.
 - If `reveal` is 1, the grader will act as if Bohan tells everyone their room number.

If the array returned by `guess` matches $[P[0], P[1], \dots, P[N - 1]]$, the sample grader prints

```
Accepted
```

In addition, the grader prints the function calls made in order for debugging purposes.