Magic Cards

Pak Dengklek will play a magic trick. Pak Dengklek's assistant, Pak Ganesh, has N cards numbered from 1 to N. A spectator is invited to the stage to choose K distinct cards out of them and give them to Pak Ganesh. Pak Ganesh sees the card, then discard one of the K cards, then leaves the remaining $K-1$ cards in some order on the table. Pak Dengklek then looks at the $K-1$ cards on the table and must be able to determine the card discarded by Pak Ganesh.

Obviously, Pak Dengklek and Pak Ganesh must not communicate right after the trick is started, but they can determine their strategy before the trick is started. You must help them by designing their strategy. This time, Pak Dengklek and Pak Ganesh will play this trick Q times with the same value of N and K.

Implementation Details

You should implement the following procedures:

```
void init_assistant(int N, int K)
```

- N: the number of cards in the trick.
- K: the number of cards chosen by the spectator.
- This procedure is called exactly once, before any calls to `choose_cards`.

```
int[] choose_cards(int[] cards)
```

- $cards$: an array of size K, consisting of the card numbers chosen by the spectator in increasing order.
- This procedure should return the $K-1$ cards left by Pak Ganesh on the table together with the order. All elements must be unique and exist in the `cards` array.
- This procedure is called exactly Q times.

```
void init_magician(int N, int K)
```

- N: the number of cards in the trick.
- K: the number of cards chosen by the spectator.
- This procedure is called exactly once, before any calls to `find_discarded_card`.

```
int find_discarded_card(int[] cards)
```

- $cards$: an array of size $K-1$ consisting of the card numbers left on the table in that order.
- This procedure should return the card number discarded by Pak Ganesh.
- This procedure is called exactly Q times.

Each test case involves a single scenario of N and K. A program that calls the above procedures is run exactly two times, as follows.

During the first run of the program:

- `init_assistant` is called exactly once before any calls to `choose_cards`;
- `choose_cards` is called exactly Q times. In each call, the returned chosen cards are stored in the grading system.

During the second run of the program:

- `init_magician` is called exactly once before any calls to `find_discarded_card`;
- `find_discarded_card` is called exactly Q times. In each call, an **arbitrary** play of the trick is chosen, and the cards returned by `choose_cards` are used as the inputs to `find_discarded_card`.

In particular, any information saved to static or global variables in the first run of the program is not available in the second run of the program.

Example

Consider the following call:

```
init_assistant(5, 3)
```

There are 5 cards that will be used in all tricks, each will invite a spectator to choose 3 distinct cards.

After initialization has been done by Pak Ganesh, consider the following call:

```
choose_cards([1, 2, 3])
```

This means the spectator chose cards numbered 1, 2, and 3. Assume Pak Ganesh discarded card number 1 and left card number 3 before card number 2 on the table, then `choose_cards` should return $[3, 2]$.

Consider another possible call:

```
choose_cards([1, 3, 4])
```

This means the spectator chose cards numbered 1, 3, and 4. Assume Pak Ganesh discarded card number 3 and left card number 1 before card number 4 on the table, then `choose_cards` should return $[1, 4]$.

Assume Pak Ganesh has left the cards on the table for all plays and consider the following call:

```
init_magician(5, 3)
```

The same information of N and K as Pak Ganesh is given to Pak Dengklek.

After initialization has been done by Pak Dengklek, consider the following call:

```
find_discarded_card([1, 4])
```

This means Pak Dengklek sees card numbers 1 and 4 in that order on the table. These cards are the same as the return value of `choose_cards([1, 3, 4])`. As Pak Ganesh discarded card number 3 in that play, then `find_discarded_card` should return 3.

Consider another call:

```
find_discarded_card([3, 2])
```

This means Pak Dengklek sees card numbers 3 and 2 in that order on the table. These cards are the same as the return value of `choose_cards([1, 2, 3])`. As Pak Ganesh discarded card number 1 in that play, then `find_discarded_card` should return 1.

Constraints

- $2 \leq K \leq 8$
- $K \leq N \leq 10\,000$
- $1 \leq Q \leq 50\,000$

For each call to `choose_cards`:

- $1 \leq cards[i] \leq N$ (for all $0 \leq i \leq K - 1$).
- All the elements of $cards$ are distinct.

For each call to `find_discarded_card`:

- All the inputs given are the same as all Q return values of `choose_cards` in random order.

Subtasks

1. (5 points) $N \leq 3$, $K = 2$.
2. (11 points) $N \leq 5$, $K = 3$.
3. (24 points) $N \leq 12$, $K = 6$.
4. (60 points) $N \leq 10\,000$, $K = 8$.

Sample Grader

The sample grader reads the input in the following format:

- line 1: N K Q
- line $2 + i$ ($0 \leq i \leq Q - 1$): the K cards chosen by the spectator in the i-th play in increasing order.

For each play, if the trick is played correctly, the sample grader prints `Accepted: chosen_cards = [chosen_cards]; discarded_card = [discarded_card]`, where `[chosen_cards]` is the cards returned by `choose_cards` and `[discarded_card]` is the card returned by `find_discarded_card`.

For each play, if the trick is failed to be played correctly, the sample grader prints `Wrong Answer: MSG`. The meaning of `MSG` is as follows:

- `invalid number of chosen cards`: the number of cards returned by `chosen_cards` is incorrect.
- `invalid chosen card number`: any of the card numbers returned by `chosen_cards` is invalid.
- `duplicated chosen cards`: there exist two cards returned by `chosen_cards` with the same number.
- `wrong discarded card`: the card returned by `find_discarded_card` is not correct.