Task 5: `Shuffle`

Note that only C++ and Java may be used for this task.

Lim Li loves anime, and has recently bought a complete season of Umaru from Amazon. The N episodes of Umaru comes in a set of N CDs, with one episode in each CD. The CDs are labelled $1, \ldots, N$, and the episodes of Umaru are also numbered $1, \ldots, N$. She was prepared to begin watching, when she realised that due to a manufacturing fault, the episode numbers don't match up with the labels on the CDs!

She immediately contacts Amazon, and is informed that although the CD labels do not match up with the episode numbers, the episodes are merely shuffled and **no single episode is missing from the set**. Wishing to avoid spoilers, Lim Li is unwilling to play the CDs herself to figure out which episode lies in each CD.

Lim Li thus decides to engage the help of her friend, Rar the Cat. The procedure she will follow is as follows:

- Lim Li will group her CDs into B boxes, containing K CDs each. ($N = B \times K$)
- The boxes are shipped to Rar. In transit, the CDs in each box may move around within the box and the boxes themselves may arrive in a different order. As such, the order of CDs in each box and the order of the boxes themselves may be shuffled in transit.
- Rar will receive the boxes, and play the CDs on his own CD player.
- Rar will then write down the set of episodes he finds in each box, and send this information back to Lim Li in B separate pieces of paper. Each piece of paper states K integers, the episode numbers of the CDs inside the box.
- The entire set of N CDs are returned to Lim Li.

This procedure can be done a maximum of Q times, before Rar gets annoyed and stops helping her. Help Lim Li figure out the episode numbers in each CD with minimal help from Rar.

Implementation Details

This is an interactive task. Do not read from standard input or write to standard output.

You are to implement the following function:

- C++: `vector<int> solve(int N, int B, int K, int Q, int ST)`
- Java: `public int[] solve(int N, int B, int K, int Q, int ST)`

The `solve` function will be called exactly once.

You are allowed to call the following grader functions to complete the task:

- C++: `vector<vector<int>> shuffle(vector<vector<int>> boxes)`
- Java: `public static int[][] shuffle(int[][] boxes)`

The function `shuffle` is called with a 2-dimensional array `boxes`, describing the grouping of the N CDs into B boxes by Lim Li. `boxes` is to be formatted such that it contains B 1-dimensional arrays, each of which contains K integers representing the labels of the CDs in the box. If your program calls this function more than Q times or with invalid parameters, the program will terminate immediately and you will be given a *Wrong Answer* verdict.

The array passed to the function `shuffle` will **not** be modified. In other words, you can expect the 2D array `boxes` to remain the same after calling the function `shuffle`.

The function `shuffle` will return a 2-dimensional array, describing the information as given by Rar. The 2-dimensional array consists of B 1-dimensional arrays, each contains K integers describing the episode numbers of the CDs in the box.

Sample Interaction

Suppose $N = 6, B = 3, K = 2, Q = 100$, and the testcase belongs to Subtask 2. The episodes are `[3, 1, 4, 5, 2, 6]`. Your function will be called with the following parameters:

```
solve(6, 3, 2, 100, 2)
```

A possible interaction could be as follows:

- `shuffle([[1, 2], [3, 4], [5, 6]]) = [[6, 2], [5, 4], [3, 1]]`

  The 2D array [[1, 2], [3, 4], [5, 6]] represents the CD labels in the three boxes sent by Lim Li. Since the boxes are mis-ordered and the CDs in each box are shuffled, the boxes received by Rar may correspond to the 2D array [[6, 5], [4, 3], [1, 2]]. Rar finds the episodes for these CD labels and reports [[6, 2], [5, 4], [3, 1]].

- `shuffle([[2, 6], [3, 1], [5, 4]]) = [[6, 1], [2, 5], [4, 3]]`

  Since the boxes are shuffled, the boxes received by Rar may correspond to the 2D array [[6, 2], [5, 4], [3, 1]]. Rar finds the episodes for these CD labels and reports [[6, 1], [2, 5], [4, 3]].

- `shuffle([[6, 5], [4, 2], [3, 1]]) = [[5, 1], [3, 4], [2, 6]]`

  Since the boxes are shuffled, the boxes received by Rar may correspond to the 2D array [[4, 2], [1, 3], [5, 6]]. Rar finds the episodes for these CD labels and reports [[5, 1], [3, 4], [2, 6]].

At this point, your program decides that it has enough information and concludes that the episodes are `[3, 4, 1, 5, 2, 6]`. As such, your program will return `[3, 4, 1, 5, 2, 6]`. As the episodes are correct, and the program has used less than 100 queries, it would be deemed as correct for this testcase.

Subtasks

The maximum execution time on each instance is 1.0s. For all testcases, the input will satisfy the following bounds:

- $B, K \geq 2$
- $N \geq 6$
- $N = B \times K$

Your program will be tested on input instances that satisfy the following restrictions:

Subtask	Marks	Additional Constraints
1	2	$N = 6, B = 2, K = 3, Q = 100$
2	3	$N = 6, B = 3, K = 2, Q = 100$
3	12	$N \leq 1000, Q = 12$. The order of the B boxes will remain the same.
4	16	$N \leq 1000, K = 2, Q = 4$
5	15	$N \leq 1000, B = 2, Q = 12$
6	52	$N \leq 1000, Q = 2000$, $B, K > 2$. Please read ”Scoring” for further details.

Scoring

Subtask 6 is a special scoring subtask. Your score depends on the maximum number of queries q you make in any testcase.

- If $q > 2000$, you will score 0 points.
- If $500 < q \leq 2000$, you will score 8 points.
- If $50 < q \leq 500$, you will score 17 points.

- If $9 < q \leq 50$, you will score $22 + 30 * (\frac{50-q}{41})^2$ points.
- If $q \leq 9$, you will score 52 points.

Testing

You may download the grader file, the header file (for C++) and a solution template under *Attachments*. The sample testcases are also provided for your reference. Please use the compile and run scripts provided (the .sh files) for testing.

Grader Input Format for Testing

- one line with five integers, N, B, K, Q and ST, where ST is the subtask number.
- one line with N integers, the episodes, $E_1, E_2, ..., E_N$