*K*TH NUMBER

Congratulations on being chosen as a contestant on the hit game show *Kth Number*!

On Kth Number, the host chooses three integers N, M, and K, where $2 \leq M < N$ and $1 \leq K \leq M$. Additionally, the host chooses a *permutation* $P = [P_0, \ldots, P_{N-1}]$, which is an array of N integers where each integer from 1 to N appears exactly once.

The host tells you N and M, but K and the permutation P are kept secret. Your goal is to find the value of K. To do this, you can ask the host a particular type of question. In each question, you select M distinct indices $I_0, \ldots, I_{M-1}$, and the host will tell you the Kth smallest value in the array $[P_{I_0}, \ldots, P_{I_{M-1}}]$.[1]

For example, imagine that the host chose $N = 5$, $M = 3$, $K = 2$, and $P = [3, 1, 5, 4, 2]$. The host tells you that $N = 5$ and $M = 3$, but K and P are secret:

- If you ask a question with $I = [1, 2, 3]$, then the host will tell you the Kth smallest number in $[P_1, P_2, P_3] = [1, 5, 4]$, which is 4.
- If you ask a question with $I = [1, 4, 0]$, then the host will tell you the Kth smallest number in $[P_1, P_4, P_0] = [1, 2, 3]$, which is 2.

You are scored based on the number of questions that you use to find K. See the scoring section for details.

Each test case has T independent scenarios. You must correctly solve all of them to solve the test case.

Implementation Details

In this task, **do not read from standard input nor write to standard output.** Do not interact with any files. Do not implement a `main` function. Instead, begin your program by including the header file `kthnumber.h` (`#include "kthnumber.h"`) and interact with it as described below.

Functions

You must implement the function `find_K`, which is called T times in each test case:

```
int find_K(int N, int M, int subtask);
```

- `N` and `M` are the values described above.
- `subtask` is the subtask number.
- From this function you may call `ask_host`.
- This function must return K.
- This function will be called T times in a single test case. We recommend resetting any global variables each time it is called.

From `find_K`, you can make calls to `ask_host`:

```
int ask_host(std::vector<int> I);
```

- The size of `I` must be M.
- It must hold that $0 \leq I_i \leq N - 1$ for all i, and $I_i \neq I_j$ for all $i \neq j$.
- This function will return the Kth smallest value in $[P_{I_0}, \ldots, P_{I_{M-1}}]$.

[1] For example, if $K = 1$, then the Kth smallest value in an array is the smallest value in that array. If $K = 2$, then the Kth smallest value is the 2nd smallest value.

- Your score depends on the number of calls made to this function. See the scoring section for details.

If you violate any of the conditions listed above, your program will be judged as incorrect and you will receive 0% of the points for the test case.

Subtasks and Constraints

For all subtasks:

- $1 \le T \le 150$, where T is the number of scenarios in a single test case.
- $3 \le N \le 200$ and the sum of N across all scenarios in a single test case is at most 500.
- $2 \le M \le N - 1$.
- $1 \le K \le M$.
- P is a permutation of length N. That is, $1 \le P_i \le N$ for all i and $P_i \neq P_j$ for all $i \neq j$.

Additional constraints for each subtask are given below.

Subtask	Points	Additional constraints
1	25	$M = 2$.
2	30	$M = N - 1$.
3	45	No additional constraints.

Scoring

If your solution violates any of the conditions in the Implementation Details section, or if you fail to correctly find K, your score will be 0.

Otherwise, let Q be the number of calls that your solution makes to `ask_host` in a single scenario. Your score depends on Q. Note that $\lceil x \rceil$ denotes the ceiling of x, which is x rounded up to the nearest integer. So, for example, $\lceil 2.1 \rceil = \lceil 2.8 \rceil = \lceil 3 \rceil = 3$.

- In Subtask 1:
  - If $Q \le 3$, your score will be 25. Otherwise,
  - If $Q \le \lceil \frac{N}{2} \rceil$, your score will be 15. Otherwise,
  - If $Q \le N^2$, your score will be 10. Otherwise,
  - Your score will be 0.
- In Subtask 2:
  - If $Q \le \lceil \frac{N}{2} \rceil + 1$, your score will be 30. Otherwise,
  - If $Q \le N$, your score will be 15. Otherwise,
  - Your score will be 0.
- In Subtask 3:
  - If $Q \le \lceil \frac{N}{2} \rceil + 1$, your score will be 45. Otherwise,
  - If $Q \le \lceil 0.8N \rceil$, your score will be 40. Otherwise,
  - If $Q \le N$, your score will be 30. Otherwise,
  - If $Q \le NM$, your score will be 20. Otherwise,
  - Your score will be 0.

Your score for a test case will be the **minimum** score of all scenarios in the test case. Your score for a subtask will be the **minimum** score of all test cases in the subtask.

Experimentation

In order to experiment with your code on your own machine, first downloaded the provided files `kthnumber.cpp`, `kthnumber.h` and `grader.cpp`.

You should modify `kthnumber.cpp`, which contains a basic example implementation.

Compile your solution with:

```
g++ -std=c++20 -O2 -Wall kthnumber.cpp grader.cpp -o kthnumber
```

This will create an executable `kthnumber` which you can run with `./kthnumber`. If you have trouble compiling, please send a message in the Communication section of the contest website.

The provided grader reads from standard input in the following format:

- The 1st line of input contains T and *subtask*.
- T scenarios follow:
  - The 1st line of each scenario contains the integers N, M, and K.
  - The 2nd line of each scenario contains the N integers $P_0, \ldots, P_{N-1}$.

The sample grader will print whether your solution correctly solves each scenario, and the score it would earn.

Please note that the grader that is used on the judging machine may have **different** behaviour to the provided grader. In particular, the grader on the judging machine may adapt the host's responses depending on your questions, rather than using a fixed permutation P and integer K.

Sample Grader Input & Sample Session

The sample grader is supplied with the following input:

```
1 3
5 3 2
3 1 5 4 2
```

One possible interaction is described below:

Grader	Student	Description
`find_K(6, 1)`		The grader calls your program.
	`ask_host({1, 2, 3})`	You ask a query with $I = [1, 2, 3]$.
returns `4`		$[P_1, P_2, P_3] = [1, 5, 4]$ and the $K = 2$nd smallest number is 4.
	`ask_host({1, 4, 0})`	You ask a query with $i = 4$, $j = 5$, and $k = 6$.
returns `2`		$[P_1, P_4, P_0] = [1, 2, 3]$ and the $K = 2$nd smallest number is 2.
	returns `2`	You return $K = 2$. This is correct.

In this example, you have solved the test case correctly using $Q = 2$ questions. This would earn you 100% of the points.

Additional Sample Grader Input

Two more sample grader inputs are supplied below, for subtasks 1 and 2.

Sample Grader Input 2

```
4 1
3 2 1
1 2 3
3 2 2
1 2 3
6 2 1
4 5 6 1 2 3
6 2 2
4 5 6 1 2 3
```

Sample Grader Input 3

```
7 2
3 2 1
1 2 3
3 2 2
1 2 3
6 5 1
4 5 6 1 2 3
6 5 2
4 5 6 1 2 3
6 5 3
4 5 6 1 2 3
6 5 4
4 5 6 1 2 3
6 5 5
4 5 6 1 2 3
```