S1-C. Guess the Permutation

Time limit	1 s
Memory limit	64 MB

Description

It is the end of the semester. Ganesh and his classmates have just finished the final exam. However, Mr. Dengklek, his favorite Mathematics teacher, suddenly announces to everyone that the exam results are disappointing. As a result, Mr. Dengklek comes up with an idea to distribute additional marks for those who are able to complete his challenge. Ganesh is aware that his exam result must be pretty bad, and hence he asks for your help to complete the challenge.

Mr. Dengklek gives every student a chance to play a game called "Guess the Permutation". Mr. Dengklek has a secret permutation $p_1, p_2, ..., p_N$ which is a permutation of $[1, 2, ..., N]$. The task is to guess the secret permutation by using operation `query(q_1, q_2, ... q_N)` where $q_1, q_2, ..., q_N$ is a permutation of $[1, 2, ..., N]$; then Mr. Dengklek will process the query and return the number of positions i for which $p_i = q_i$. Mr. Dengklek does not enjoy answering the query; hence, the more queries asked, the lower the score.

Help Ganesh to guess the secret premutation p using small enough number of queries.

Implementation details

You should implement the following function:

- `void solve(int N)`
  - `N`: the length of the secret permutation p.
  - The secret permutation p consists of distinct integers from 1 to N inclusive.
  - This function is called exactly once, and should be terminated after receiving a value that is equal to N from the query function, which means you have managed to guess the permutation correctly.

The function above can make calls to the following function:

- `int query(vector<int> q)`
  - `q`: a vector of length N, describing a permutation of $1, 2, ..., N$.
  - The elements of `q` must be distinct integers from 1 to N inclusive.
  - This function returns the number of different indices i, such that $p_i = q_i$.

Example

Example 1

Consider a scenario in which $N = 3$ and the secret permutation p is $[2, 1, 3]$. The function `solve` is called in the following way:

`solve(3)`

This function may call `query([1, 2, 3])`, which (in this scenario) returns 1. It may then call `query([1, 3, 2])`, which returns 0. It may then call `query([2, 1, 3])`, which returns 3.

At this point, the secret permutation has been guessed. So, the function `solve` should be terminated.

Subtask

All subtasks have the following constraints:

- $1 \leq N \leq 256$
- $1 \leq p_i \leq N$
- $p_i \neq p_j$ for $1 \leq i < j \leq N$

Subtask 1 (13 poin)

- $1 \leq N \leq 7$

Subtask 2 (38 poin)

- $1 \leq N \leq 50$

Subtask 3 (49 poin)

- No additional constraint

Scoring

In any of the test cases, if the call to the procedure `solve` does not confirm the rules mentioned above, or if it does not terminate, the score of your solution will be 0 points.

Let Q be the number of queries used in a test. Then the scoring is as follows:

- For Subtask 1:
  - if $Q \leq 50$, then 13 points;
  - if $50 < Q \leq 200$, then 9 points;
  - if $200 < Q \leq 5040$, then 6 points;
  - if $Q > 5040$, then 0 points.
- For Subtask 2: let Q' = $(floor(Q/100) + 1) \times 100$
  - if $Q' \leq 400$, then 38 points;
  - if $400 < Q' \leq 700$, then $(38 - 29) \times (700 - Q')/(700 - 400) + 29$ points;
  - if $700 < Q' \leq 1300$, then $(29 - 21) \times (1300 - Q')/(1300 - 700) + 21$ points;
  - if $1300 < Q' \leq 10000$, then $(21 - 4) \times (10000 - Q')/(10000 - 1300) + 4$ points;
  - if $10000 < Q'$, then 0 points.
- For Subtask 3: define Q' as in Subtask 2
  - if $Q' \leq 2400$, then 49 points;
  - if $2400 < Q' \leq 5000$, then $(49 - 29) \times (5000 - Q')/(5000 - 2400) + 29$ points;
  - if $5000 < Q'$, then 0 points.

Sample grader

The sample grader reads a vector p which is a permutation of $[1, 2, ..., N]$. The sample grader reads input in the following format:

- line 1: N
- line 2: p_1 p_2 ... p_N

The output of sample grader is in the following format:

- line 1: Verdict
  - "AC" if the program manages to find the correct permutation.
  - "WA" otherwise.
- line 2: Generic string message
  - The number of calls to `query` if line 1 is "AC".
  - Otherwise, information about the issue that causes the program to output a "WA".

Notes

- The test cases are not adaptive.
- The grader that is given to participants is different from the one that is being used by jury.
- Participants should terminate the call to function `solve(N)` gracefully (i.e., using `return` statement or by reaching the end of the function). Using other methods such as `exit(0)` could cause the submission to be considered as "Wrong Answer" by the grading system.
- If the correct permutation is found after too many queries, you will get the "WA" verdict and 0 points.
- Your score for each subtask is the minimum score among all the results on tests of the corresponding subtask.