COLEGIUL NAȚIONAL DE INFORMATICĂ TUDOR VIANU

ROMANIAN MASTER OF INFORMATICS 2019

Secret Permutation

The Scientific Committee has hidden from you a permutation `P` of all the integers from `1` to `N`. (`3 ≤ N ≤ 256`). You need to find it. Permutation `P` is fixed (the grader is not adaptive).

In your endeavor, you are allowed to ask queries that take as parameter another permutation `V` of all the integers from `1` to `N`:

`query(V)` will return `sum(i = 1..N - 1, abs(P[V[i]] - P[V[i + 1]]))`.

Performing a number of queries, you are to discover permutation `P`, **or any other permutation** `P'` that is indistinguishable from `P`. Two permutations are indistinguishable if queried in all possible ways they both yield the same answers.

Interaction

This is an interactive problem. You must submit a source file with the following constraints:

C / C++: `#include "permutationc.h"`	You **must** include this header file in order to properly compile your code and link it with the Scientific Committee's code.
C / C++: `void solve(int N);`	Your solution to this problem must be written inside this function. You are free to write and call additional functions but you're **not allowed** to write a `main()` function.

C / C++:
`int query(int V[]);`
or C++ only:
`int query(std::vector<int> V);`
	Whenever you want to perform a query, call this function with a permutation `V` of all the integers from `1` to `N` as parameter. You will be graded based on the number of times you call this function.

C / C++:
`void answer(int P[]);`
or C++ only :
`void answer(std::vector<int> P);`
	When you're confident you've discovered permutation `P`, call this function with `P` as a parameter. Calling this function will **terminate** the program.

Note that both permutations `P` and `V` are represented as a `0`-indexed `int array` or `std::vector<int>` when supplied as parameters.

Example

Sample code to illustrate the Interaction section:

C:

```
#include "permutationc.h"

void solve(int N) {
  if (N == 2) {
    int V[] = {1, 2};
    int qAns = query(V);
    if (qAns == 1) {
      int P[] = {1, 2};
      answer(P);
    }
  }
}
```

C++:

```
#include "permutation.h"

void solve(int N) {
  if (N == 2) {
    std::vector<int> V = {1, 2};
    int qAns = query(V);
    if (qAns == 1) {
      std::vector<int> P = {1, 2};
      answer(P);
    }
  }
}
```

Sample grader

For local testing you can download two files from CMS: **`sample_grader.cpp`** and **`permutation.h`**.

The Grader reads from Standard Input an integer **`N`** - the size of the hidden permutation and **`N`** distinct integers - the hidden permutation. Then, the Grader calls the **`solve()`** function you must implement.

At Standard Output the Grader will output:

(a) for every **`query()`** call: the queried permutation and the answer to the query;

(b) for the **`answer()`** call: the verdict (**`Correct`** or **`Wrong Answer`**), N and Q - the size of the permutation and the number of queries you used.

Subtasks

(1) **$3 \le N \le 7$ (15 points)**

(2) **$3 \le N \le 50$ (35 points)**

(3) **$3 \le N \le 256$ (50 points)**

Scoring

Each of the test cases is scored as follows:

If you fail to discover one of the correct permutations, then `0%` of the score is awarded.

Otherwise, let `Q` be the number of queries you needed to solve the test case.

(a) If `Q ≤ N` then `100%` of the score is awarded.

(b) If `N ≤ Q ≤ 2 * N` queries then `(100 - 40 * (Q - N) / N)%` (between `60%` and `100%`, increasing as `Q` decreases) of the score is awarded.

(c) If `2 * N ≤ Q ≤ N^2` queries then `(60-40 * (Q - 2 * N) / (N^2 - 2 * N))%` (between `20%` and `60%`, increasing as `Q` decreases) of the score is awarded.

(d) If `N^2 ≤ Q` then `20%` of the score is awarded.

The total score of this task will be rounded to `2` decimal places.

The Scientific Committee has a solution scoring over `98` points.