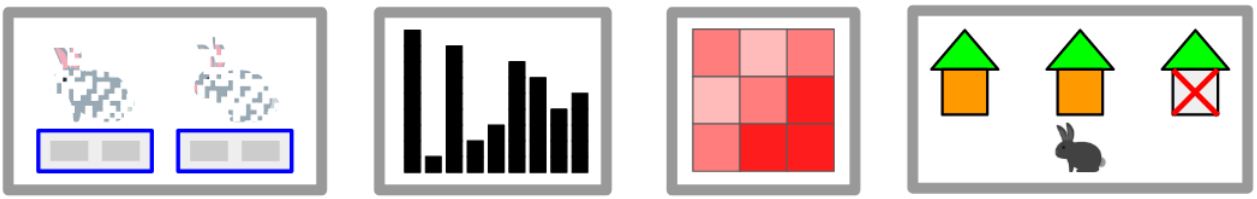
4 Artswap

Time Limit: 1.0s
Memory Limit: 256MB

4.1 Problem Description

Whiterabbit has recently procured some famous art pieces for display in the Bunnyland Art Museum. He has been able to get N art pieces in total, and the museum has set aside a long hallway to display these art pieces; this hallway has exactly N positions to put up the art pieces, numbered 1 to N from left to right.

As the workers in charge of transporting the art pieces were not given specific instructions on how to place the art pieces, the pieces have been placed in some arbitrary manner, with 1 art piece per position. The art piece at the i^{th} position has grade G_i, which is a distinct grade from 1 to N based on its history, critique from art critics, number of rabbits in the art piece etc.



The museum has also been outfitted with an art scanner, which is meant to help Whiterabbit figure out the engagement value of the arrangement of art pieces. To be exact, the engagement value of the art pieces with grades $\{G_1, G_2, \cdots, G_N\}$ in this order is equal to the number of pairs (i, j) such that $i < j$ but $G_i > G_j$. The initial engagement value of the current arrangement is E.

Unfortunately, Whiterabbit needs to know the exact arrangement of the art pieces so that he can properly plan the exhibit. He can't go to the museum as it is too late, but he can still contact the workers at the museum. The hallway is too long for the workers to figure out where each art piece is, but they can help swap some art pieces.

More specifically, Whiterabbit can give the workers **2 distinct grades (not positions)** A and B, and the workers will swap the positions of the 2 art pieces with grades A and B. Whiterabbit can then use the scanner to find the engagement value of the arrangement of art pieces.

Whiterabbit can't keep asking the workers to swap the artworks forever though; he can only ask them to swap the artworks Q times before they get too tired. Help Whiterabbit determine the original arrangement of the artworks **AND** return all of the artworks to their original positions using at most Q swaps.

4.2 Interaction Protocol

This is a function call problem. You need to add `#include "artswap.h"` at the start of your program, and implement the following function:

- `void artswap(int N, long long E, int S, int Q)`
  - N is the number of positions and art pieces.
  - E is the initial engagement value.
  - S is the subtask this testcase belongs to.
  - Q is the maximum number of swaps Whiterabbit can ask for.
  - This function will be called exactly once per testcase, and should make use of the `swap_pieces` and `answer` functions.

You should make use of the following grader functions:

- `long long swap_pieces(int X, int Y)`
  - X and Y are grades of the 2 art pieces you want to swap. X and Y should lie between 1 and N inclusive.
  - This function returns the new engagement value of the arrangement of art pieces after the 2 given art pieces have swapped positions.
  - This function can be called at most Q times.
  - The final arrangement of the art pieces should be exactly the same as when it started.
- `void answer(int P, int G)`
  - P is the position that you want to provide the answer for.
  - G is what you think the grade of the art piece was in the initial arrangement.
  - This function should be called exactly N times, one for each position from 1 to N.
  - The grades for all N positions should be exactly the same as the initial arrangement.

If any of the above conditions are not satisfied, your program is judged as Wrong Answer. Otherwise, your program is judged as Accepted and your score is calculated by the number of calls to `swap_pieces` (see Subtasks).

4.3 Subtasks

Subtask	Score	N	Q	Additional Restrictions
1	7	$N = 2$	$Q = 0$	
2	16	$N = 8$	$Q = 100000$	
3	15	$N = 400$	$Q = 100000$	$E \leq 2$
4	48	$N = 400$	$Q = 100000$	
5	14	$N = 50000$	$Q = 100000$	
For all subtasks: $2 \leq N \leq 50000$				

Subtasks 4 and 5 are partial scoring subtasks. For each testcase, let X be the number of `swap_pieces` queries that your program uses.

1. If $16N < X$, your score is $20 + 20\left(1 - \log_2\left(\frac{X+168N}{184N}\right)\right)$
2. If $2N < X \leq 16N$, your score is $40 + 30\left(\frac{16N-X}{14N}\right)$
3. If $2N - 2 < X \leq 2N$, your score is 70.
4. If $2N - 4 < X \leq 2N - 2$, your score is 80.
5. If $X \leq 2N - 4$, your score is 100.

The scores for Subtask 4 and 5 is the **average** across all testcases in that subtask multiplied by the points allocated for that subtask. Take note that (1) and (2) do not apply for subtask 5.

4.4 Sample Grader

In the Attachments, you can find a sample grader `grader.cpp`. This grader is similar **but not exactly identical** to the one used to grade your submitted program. It takes in input in the following format:

1. The first line should contain 3 spaced integers, N, S and Q.
2. The next line should contain N spaced integers, the i^{th} one being G_i.

The grader will then call the function `artswap` and count the number of queries your program takes. The grader could give one of the following 5 "Wrong Answer" verdicts:

1. Invalid swap attempted
2. Invalid answer attempted
3. Too many swaps made
4. Answer arrangement not equal to original
5. Final arrangement not equal to original

If your program is able to provide the correct arrangement and return the art pieces to their original arrangement within the query limit, the grader will output "Correct" followed by "Queries Used : X" where X is the number of calls to `swap_pieces` that your program uses (both without quotes). The amount of time needed for the sample grader to compute the answer after every `swap_pieces` is likely to be much more than what the real grader will take.

Note: The time limit for this problem is 1.0 seconds. However, the time limit for subtask 5 is 3.0 seconds to account for the time the grader takes to answer all of the queries.

A compile command to compile your program (`artswap.cpp`) with the grader has been included in the Attachments.

4.5 Example

Consider the case of $N = 4$, $S = 0$, $Q = 40000$, and $G = [2, 4, 1, 3]$. This testcase corresponds to `sample-01.txt`. In this case, the initial engagement value is 3 due to the pairs of indices $(1, 3)$, $(2, 3)$ and $(2, 4)$. Notably, $S = 0$ here represents a sample testcase.

Sample Calls			
Grader Call	Answer Call	Return	G
`artswap(4,3,0,40000)`			2 4 1 3
	`swap_pieces(4,2)`	4	4 2 1 3
	`swap_pieces(2,4)`	3	2 4 1 3
	`swap_pieces(3,2)`	4	3 4 1 2
	`swap_pieces(3,4)`	5	4 3 1 2
	`swap_pieces(2,4)`	2	2 3 1 4
	`swap_pieces(3,4)`	3	2 4 1 3
	`answer(1,2)`		2 4 1 3
	`answer(2,4)`		2 4 1 3
	`answer(3,1)`		2 4 1 3
	`answer(4,3)`		2 4 1 3

Once `artswap` has terminated, the grader checks that both the answers given and the current arrangement match the original arrangment. Since both are correct and the number of queries to `swap_pieces` (6) is less than $Q = 40000$, the grader will output "Correct" followed by "Queries Used: 6" (without quotes).

Note that another sample testcase `sample-02.txt` has also been provided in the problem attachments.