Problema 1. Mole

Fișier header:	`mole.h`
Limită de timp:	0.25 secunde
Limită de memorie:	1024 megabytes

Pentru a nu mai întâmpina probleme logistice, comisia lotului de pregătire de anul acesta a decis să stabilească clasamentul celor N participanți independent de punctajele obținute la cele două probe de concurs.

Din fericire pentru tine, însă, există o **cârtiță** (*eng. mole*), a cărui nume nu îl vom menționa, care ți-a divulgat acest secret, astfel că ai aflat intențiile malefice ale comisiei. Mai mult, persoana este dispusă să te ajute să descoperi clasamentul. Totuși, pentru a nu fi descoperiți, ați hotărât să comunicați astfel:

1. Vei întreba cârtița un posibil clasament $p_1\ p_2 \cdots p_N$;
2. Cârtița va porni de la clasamentul întrebat de tine și îl va transforma în clasamentul comisiei, interschimbând pe rând locurile a câte doi participanți. Să presupunem că numărul minim de schimbări este x. Cârtița îți va răspunde întrebării cu x.

Deoarece fiecare întrebare pusă constituie un risc ca persoana să fie descoperită, vă doriți ca întrebările puse între voi să nu fie foarte multe (vezi secțiunea **Punctare**), dar, bineînțeles, să descoperi clasamentul stabilit de comisie în cele din urmă.

Interacțiune

Această problemă este interactivă. În fișierul `mole.h` este definită funcția cu următorul antet:

```
int ask(std::vector<int> guess)
```

Atenție! Această funcție nu trebuie implementată de către voi. Graderul va implementa această funcție. Argumentul `guess` reprezintă clasamentul de care veți întreba. Funcția va returna numărul minim de intershimbări x, descris mai sus. Argumentul `guess` trebuie să reprezinte o permutare validă a numerelor din intervalul $[1, N]$. în caz contrar, graderul va termina programul din execuție și va nota testul ca fiind incorect.

Detalii de implementare

Veți implementa funcția cu următorul antet:

```
std::vector<int> find_standings(int N)
```

Funcția `find_standings` va fi apelată o dată. N este numărul de concurenți.

Funcția trebuie să returneze, în final, clasamentul căutat. Pentru aceasta, din cadrul ei se poate apela de un număr limitat de ori funcția de interacțiune `ask` (vezi secțiunea **Punctare**).

Punctare

Subtask	Punctaj	Constrângeri
1	7 puncte	$3 \leq N \leq 6$ Funcția `ask` poate fi apelată de cel mult 1 000 de ori.
2	14 puncte	$3 \leq N \leq 100$ Funcția `ask` poate fi apelată de cel mult 5 000 de ori.
3	40 de puncte	$3 \leq N \leq 200$ Funcția `ask` poate fi apelată de cel mult 4 000 de ori.
4	16 puncte	$3 \leq N \leq 200$ Funcția `ask` poate fi apelată de cel mult 2 700 de ori.
5	15 puncte	$3 \leq N \leq 200$ Funcția `ask` poate fi apelată de cel mult 1 800 de ori.
6	8 puncte	$3 \leq N \leq 200$ Funcția `ask` poate fi apelată de cel mult 1 600 de ori.

Model de grader

Graderul va începe prin a citi de la consolă datele de intrare în următorul format:

- linia 1: N, reprezentând numărul de paricipanți
- linia 2: $s_0\ s_1 \cdots s_{N-1}$ (separate prin câte un spațiu), reprezentând clasamentul secret

Pentru fiecare apel al funcției `ask`, graderul va afișa la consolă argumentul funcției, urmând să citească de la consolă răspunsul întrebării.

Dacă programul vostru va termina execuția cu succes, graderul va afișa:

- mesajul `OK [Q]`, unde Q este numărul de întrebări puse, dacă soluția returnată este corectă,
- mesajul `WA`, dacă soluția returnată este greșită.

Exemplu

intrare	ieșire

`interact(3)`
 	
`2`
 	
`1`
 	
`0`
 	
`OK 3`
	
	`ask({1, 2, 3})`
	
	`ask({1, 3, 2})`
	
	`ask({2, 3, 1})`
	
	`return {2, 3, 1}`