Dobro i zło

Dostępna pamięć: 64MB

Walka dobra ze złem w Vilolandzie trwa w najlepsze. Niedługo rozpęta się bitwa, która zadecyduje o wszystkim, tylko... no właśnie... gdzie mogłaby się odbyć?

Viloland, jak wiadomo, jest światem w kształcie prostokąta podzielonego na 1 x n kwadratowych pól. Każde pole jest kontrolowane albo przez dobro, albo przez zło, przy czym wiadomo, że pole numer 1 jest kontrolowane przez dobro, a pole numer n przez zło. To, przez kogo jest kontrolowane dane pole, może się dowolnie przeplatać i każde ze „środkowych” $n - 2$ pól nie jest w żaden sposób zależne od innych.

Wielka bitwa musi odbyć się na dwóch sąsiednich polach, z których lewe (to o mniejszym indeksie) jest kontrolowane przez dobro, a prawe przez zło. Póki co przywódcy obu stron tylko się kłócą, ale w końcu bitwa musi się odbyć. Pomóż znaleźć odpowiednie miejsce!

Komunikacja

Twój program powinien używać biblioteki, która pozwala na zadawanie pytań o to, kto kontroluje poszczególne pola, oraz na udzielenie ostatecznej odpowiedzi. Aby użyć biblioteki, należy wpisać na początku programu:

```
#include "doblib.h"
```

Biblioteka udostępnia następujące funkcje:

- `long long inicjuj();`

  Inicjalizuje bibliotekę. Należy wywołać ją tylko raz, na początku programu, przed użyciem pozostałych funkcji. Zwraca liczbę n, oznaczającą liczbę pól w Vilolandzie.

- `int sily(long long x);`

  Dla każdego x spełniającego warunek $1 \leq x \leq n$ zwraca 1, jeśli pole numer x jest kontrolowane przez siły dobra, 2, jeśli przez siły zła. W szczególności `sily(1)` zwróci 1, `sily(n)` zwróci 2. **Uwaga! Możesz użyć tej funkcji co najwyżej 100 razy!**

- `void odpowiedz(long long x);`

  Udziela odpowiedzi, że na polu numer x panują siły dobra, zaś na polu numer $x + 1$ - siły zła. Wywołanie tej funkcji zakończy działanie twojego programu.

Twój program **nie może** czytać żadnych danych (ani ze standardowego wejścia, ani z plików). **Nie może** również nic wypisywać do plików ani na standardowe wyjście. Może pisać na standardowe wyjście diagnostyczne (`stderr`) - pamiętaj jednak, że zużywa to cenny czas.

Przykładowy przebieg programu

Załóżmy, że $n = 7$, a rozkład sił na kolejnych polach wygląda tak:

1121222

gdzie podobnie jak wyżej 1 oznacza kontrolę sił dobra, a 2 - sił zła. Poprawne odpowiedzi to $x = 2$ oraz $x = 4$. Interakcja programu z biblioteką może wyglądać następująco:

Wywołanie	Zwrócona wartość
`inicjuj()`	-
`sily(2)`	1
`sily(3)`	2
`odpowiedz(2)`	-

Ocenianie

Podzadanie	Ograniczenia	Punkty
1	$n \leq 10^2$	20
2	$n \leq 10^{18}$	80

Eksperymenty

W zakładce Pliki znajduje się archiwum `dob_dlazaw.zip` zawierające:

- Przykładową bibliotekę `doblib.h`, która pozwoli Ci przetestować poprawność formalną rozwiązania. Aby jej użyć, umieść ją w tym samym folderze co swoje rozwiązanie, a następnie skompiluj je. Tak otrzymany program wczytuje z wejścia liczbę *n*, a następnie ciąg *n* znaków równych ’1’ lub ’2’, w zależności od tego czy dane pole kontrolowane jest przez siły dobra, czy przez siły zła. Pamiętaj jednak, że biblioteka ta różni się od tej, na której zostanie ostatecznie ocenione twoje rozwiązanie.
- Przykładowe, błędne rozwiązanie `dob.cpp`, ilustrujące poprawną komunikację z biblioteką.
- Test przykładowy `dob0.in` w opisanym wyżej formacie przykładowej biblioteki.