D. Dekoracje Ogrodowe

Nazwa zadania	Dekoracje ogrodowe
Limit czasu	7 sekund
Limit pamięci	1 GB

Każdego dnia podczas powrotu ze szkoły do domu, Daria mija po drodze N domów ponumerowanych od 0 do $N-1$. Obecnie dom i jest zamieszkany przez mieszkańca i. By nieco urozmaicić scenerię, mieszkańcy postanowili wymienić się domami. Osoba, która wprowadzi się do i-tego domu to osoba a_i (obecnie zamieszkująca dom a_i).

Przed każdym domem w ogrodzie stoi rzeźba ptaka (orła bielika). Rzeźby mają dwa tryby, które można przestawiać: ptak ze skrzydłami rozpostartymi (tak, jakby leciał) lub zamkniętymi (tak, jakby siedział na ziemi). Mieszkańcy są bardzo stanowczy w kwestii preferowanego trybu ich ogrodowej dekoracji. W tym momencie z przodu domu i rzeźba jest w trybie preferowanym przez mieszkańca i. Mieszkaniec nie przeprowadzi się do nowego domu, jeśli rzeźba w jego nowym ogrodzie nie jest ustawiona w preferowanym trybie. Daria postanowiła, że poustawia ptasie dekoracje, by mieszkańcy mogli się przeprowadzić.

Jej plan, by to osiągnąć, wygląda następująco: za każdym razem, gdy idzie ulicą (idąc do szkoły lub wracając z niej do domu), przygląda się rzeźbom jedna po drugiej i zmienia niektórym tryb (poprzez rozpostarcie lub zamknięcie ich skrzydeł). Jej myśli są bardzo zajęte przez szkołę i dom, więc **Daria nie pamięta trybu, w jakim były ustawione ptaki podczas poprzednich spacerów**. Na szczęście zawsze ma przy sobie rozpiskę wartości $a_0, a_1, ..., a_{N-1}$, więc wie, dokąd planuje przeprowadzkę każdy z mieszkańców.

Pomóż Darii opracować strategię, dzięki której ustawi wszystkie rzeźby w domach tak, by pasowały do preferencji ich nowych mieszkańców. Daria może przejść wzdłuż drogi co najwyżej 60 razy, ale by otrzymać wyższy wynik, powinna zrobić to w mniejszej liczbie przejść.

Implementacja

Twój program będzie uruchamiany wielokrotnie.

Przy każdym uruchomieniu powinnaś wczytać na początku jedną linię wejścia zawierające liczby w i N, odpowiednio indeks przejścia ulicą oraz liczbę domów. Przy pierwszym uruchomieniu $w = 0$,

przy kolejnym uruchomieniu $w = 1$ i tak dalej (bardziej szczegółowy opis znajduje się niżej).

W kolejnej linii wejścia znajduje się N liczb całkowitych $a_0, a_1, ..., a_{N-1}$. Osoba, która przeprowadza się do domu i obecnie mieszka w domu a_i. Wartości a_i tworzą *permutację*: każda wartość od 0 do $N - 1$ występuje wśród wartości a_i dokładnie raz. Zauważ, że mieszkaniec może nie chcieć się przeprowadzać; oznacza to, że dopuszczalne jest $a_i = i$.

Mieszkańcy zmienią swoje miejsce zamieszkania tylko raz. Oznacza to, że dla ustalonego przypadku testowego wartości N oraz a_i pozostają niezmienne przy każdym uruchomieniu.

Pierwsze uruchomienie.

Podczas pierwszego uruchomienia twojego programu, $w = 0$. Podczas tego uruchomienia powinnaś wypisać na wyjście jedną liczbę całkowitą W ($0 \leq W \leq 60$) -- ile razy chcesz, żeby Daria przeszła ulicą. Twój program powinien się następnie zakończyć. Twój program zostanie uruchomiony ponownie W razy.

Kolejne uruchomienia.

Podczas kolejnego uruchomienia, $w = 1$, podczas następnego $w = 2$, i tak dalej; Podczas ostatniego uruchomienia $w = W$.

Po wczytaniu przez Ciebie wartości w, N oraz $a_0, a_1, ..., a_{N-1}$, Daria rozpoczyna swój spacer wzdłuż ulicy.

- Jeśli w jest nieparzyste, Daria idzie z domu do szkoły i mija domy w kolejności $0, 1, ..., N - 1$.

Twój program powinien następnie wczytać linię z wartością b_0, reprezentującą aktualny tryb rzeźby ptaka w ogrodzie domu 0 -- będzie to wartość 0 (zamknięte skrzydła) lub 1 (rozpostarte skrzydła). Po wczytaniu wartości b_0, powinnaś wypisać pojedynczą linię z wartością 0 lub 1, reprezentującą nową wartość b_0.

Następnie Twój program powinien wczytać linię zawierającą wartość b_1, reprezentującą aktualny tryb rzeźby ptaka w ogrodzie domu 1 i wypisać nową wartość b_1. Tak postępuj dla każdego z N domów. Po tym, jak Daria minie ostatni z domów (to znaczy, Ty wczytasz aktualną i wypiszesz nową wartość b_{N-1}), Twój program powinien się zakończyć.

Zauważ, że Twój program może wczytać wartość b_{i+1} dopiero po wypisaniu wartości b_i.

- Jeśli w jest parzyste, Daria wraca do domu po szkole i mija domy w odwrotnej kolejności: $N - 1, N - 2, ...0$. Oznacza to, że rozpoczynasz od wczytania aktualnej i wypisania nowej wartości b_{N-1}, następnie b_{N-2} i tak dalej aż do b_0.

Dla $w = 1$, wartości $b_0, b_1, ..., b_{N-1}$ na wejściu są początkowymi trybami ptasich rzeźb. Dla $w > 1$, wartości $b_0, b_1, ..., b_{N-1}$ wczytane przez Twój program to wartości ustawione podczas

poprzedniego uruchomienia.

Na koniec, po ostatnim uruchomieniu Twojego programu, wartości b_i muszą być równe wartościom b_{a_i} dla każdego i. W przeciwnym wypadku otrzymasz rezultat Wrong Answer.

Szczegóły.

Jeśli *suma* czasów $W + 1$ uruchomień Twojego programu przekroczy limit czasu, Twoje zgłoszenie zostanie ocenione z rezultatem Time Limit Exceeded.

Upewnij się, że czyścisz bufor po wypisaniu każdej linii, w przeciwnym wypadku Twój program może zostać oceniony z rezultatem Time Limit Exceeded. W Pythonie czyszczenie bufora następuje automatycznie gdy korzystasz z `input()` do wczytywania linii wejścia. W C++, `cout << endl;` czyści bufor i wypisuje znak nowej linii; gdy używasz polecenia printf, użyj `fflush(stdout)`.

Podzadania i ocenianie

- $2 \leq N \leq 500$.
- Możesz wykorzystać co najwyżej $W \leq 60$ przejść ulicą.

Twoje rozwiązanie będzie testowane na zbiorze grup testów, każda warta pewną liczbę punktów. Każda grupa testów zawiera zbiór przypadków testowych. Żeby dostać punkty za grupę testów, musisz rozwiązać wszystkie przypadki testowe w danej grupie testów.

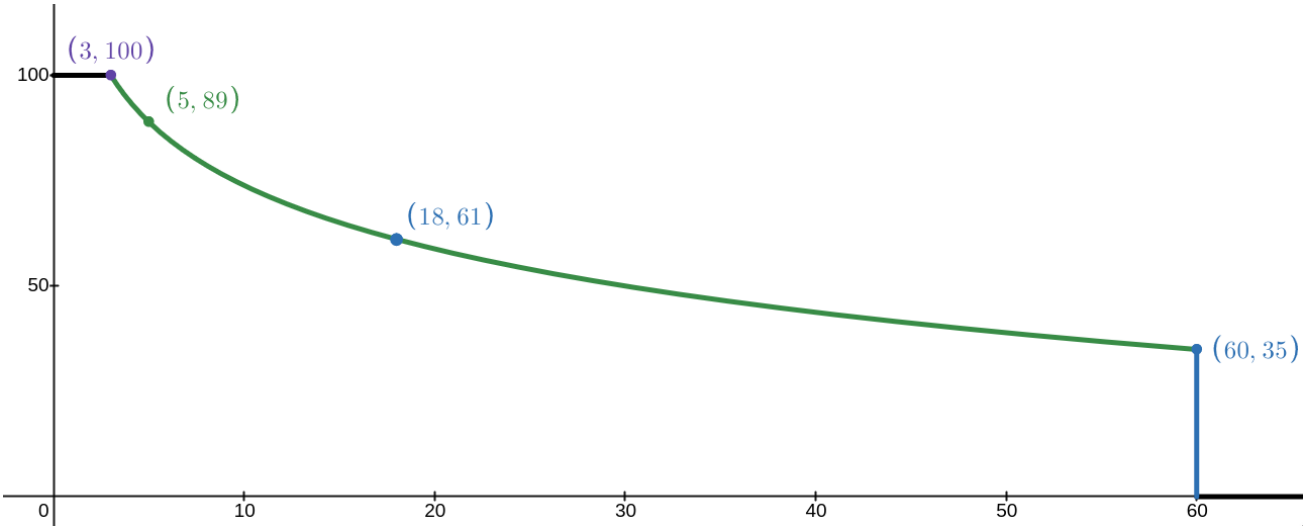
Grupa	Maksymalna punktacja	Ograniczenia
1	10	$N = 2$
2	24	$N \leq 15$
3	9	$a_i = N - 1 - i$
4	13	$a_i = (i + 1) \bmod N$
5	13	$a_i = (i - 1) \bmod N$
6	31	Bez dodatkowych ograniczeń

Dla każdej grupy testowej którą Twój program rozwiąże poprawnie, otrzymasz liczbę punktów opartą na poniższym wzorze:

$$\text{score} = S_g \cdot \left(1 - \tfrac{1}{2} \log_{10}(\max(W_g, 3)/3)\right),$$

gdzie S_g jest maksymalnym wynikiem dla tej grupy testowej, a W_g jest największą z wartości W użytych w przypadkach testowych tej grupy testowej. Twoja punktacja dla każdej grupy testowej zostanie zaokrąglona do najbliższej liczby całkowitej.

Poniższy wykres prezentuje liczbę punktów jako funkcję zależną od W, którą Twój program otrzyma, jeśli rozwiąże wszystkie grupy testowe z tą samą wartością W. W szczególności, by otrzymać wynik 100 punktów za to zadanie, musisz rozwiązać każdą grupę testową z $W \leq 3$.



Narzędzie do testowania

By wspomóc testowanie Twojego rozwiązania, udostępniamy Ci do pobrania proste narzędzie testujące. Korzystanie z narzędzia jest opcjonalne. Miej na uwadze, że oficjalna sprawdzaczka na Kattis jest inna od narzędzia testującego.

By skorzystać z narzędzia, utwórz plik wejściowy, taki jak ,,sample1.in", który zaczyna się od linii zawierającej liczbę N. W kolejnej linii powinno znajdować się N liczb reprezentujących permutację, a w kolejnej linii N bitów (0 lub 1) reprezentujących początkowy stan rzeźb orłów. Na przykład:

```
6
1 2 0 4 3 5
1 1 0 0 1 0
```

Dla rozwiązań w Pythonie, na przykład `solution.py` (uruchamianych za pomocą `pypy3 solution.py`):

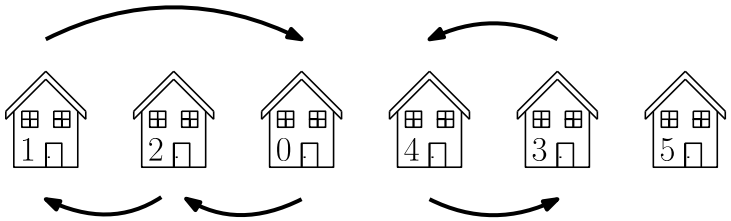
```
python3 testing_tool.py pypy3 solution.py < sample1.in
```

Dla rozwiązań w C++, skompiluj je (na przykład za pomocą komendy `g++ -g -O2 -std=gnu++20 -static solution.cpp -o solution.out`), a następnie uruchom:

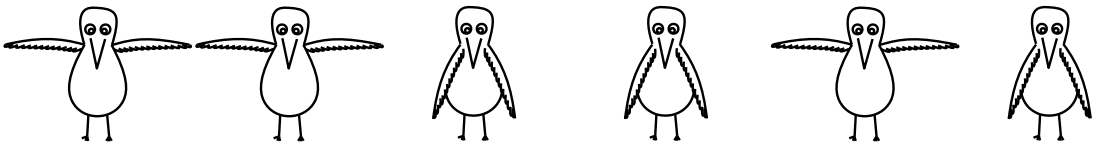
```
python3 testing_tool.py ./solution.out < sample1.in
```

Przykład

W teście przykładowym otrzymujesz poniższą permutację ludzi w domach:

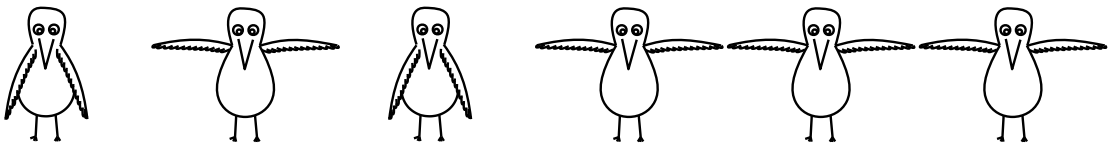


Podczas pierwszego uruchomienia przykładowy program (dla $w = 0$) wypisuje wartość $W = 2$, oznaczającą, że Daria przejdzie ulicą dwa razy (a program zostanie uruchomiony jeszcze dwukrotnie). Przed pierwszym przejściem orły w ogrodzie wyglądają tak:



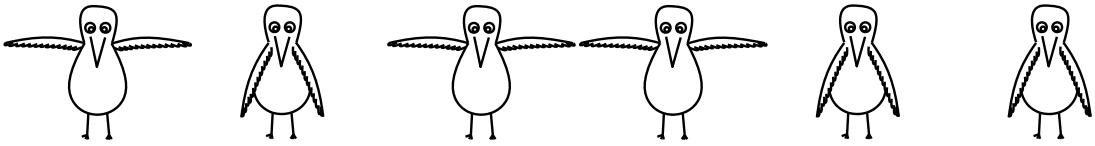
Następnie program jest uruchamiany z $w = 1$: oznaczającym pierwsze przejście Darii. Daria przechodzi wzdłuż rzeźb orłów jedna po drugiej, rozpoczynając od lewej strony i potencjalnie zmienia ich tryb. Przykładowy program musi wypisać tryb i-tego orła zanim odczyta tryb $(i + 1)$-szego orła.

Gdy Daria dociera do szkoły, orły wyglądają następująco:



Podczas ostatniego uruchomienia programu (dla $w = 2$), Daria wraca ze szkoły do domu. Pamiętaj, że w takim przypadku idzie ona wzdłuż ulicy od prawej do lewej i potencjalnie zmienia rzeźby orłów w tej kolejności! Oznacza to, że musi ustalić tryb i-tego orła zanim zobaczy $(i - 1)$-szego orła.

Po tym przejściu, orły wyglądają następująco:



Istotnie, jest to poprawne ustawienie. Na przykład, rzeźba orła 3 (to znaczy czwarta od lewej) ma rozpostarte skrzydła (obecnie $b_3 = 1$) i jest to poprawne, bo mieszkaniec 4 będzie się tam przeprowadzał ($a_3 = 4$). Ten mieszkaniec posiadał rzeźbę orła z rozpostartymi skrzydłami w swoim ogrodzie przed przeprowadzką (początkowo $b_4 = 1$).

Wyjście sprawdzaczki	Twoje wyjście
0 6	
1 2 0 4 3 5	
	2

Wyjście sprawdzaczki	Twoje wyjście
1 6	
1 2 0 4 3 5	
1	
	0
1	
	1
0	
	0
0	
	1
1	
	1
0	
	1

Wyjście sprawdzaczki	Twoje wyjście
2 6	
1 2 0 4 3 5	
1	
	0
1	
	0
1	
	1
0	
	1
1	
	0
0	
	1