D. Holuby

Problem Name	Garden Decorations
Time Limit	7 seconds
Memory Limit	1 gigabyte

Alicka každý deň chodí pešo do školy aj zo školy. Prechádza pritom vždy ulicou, na ktorej stojí v jednom rade vedľa seba N domov. Tieto domy sú za radom očíslované od 0 do $N-1$. Pre každé i platí, že v dome číslo i momentálne býva človek číslo i.

Ľudia bývajúci na ulici sa nudia, preto by si chceli medzi sebou povymieňať domy. Presnejšie, pre každé i platí, že do domu i sa nasťahuje človek s číslom a_i (teda ten, ktorý momentálne žije v dome číslo a_i).

Pred každým domom stojí jedna mechatronická soška holuba. Soška má dva možné stavy: stav 0 je, že má *zavreté* krídla (ako keď holub stojí na zemi) a stav 1 je, že má *otvorené* krídla (ako keď holub letí). Sošku sa dá prepínať z jedného stavu do druhého a späť.

Každý obyvateľ ulice je silne zvyknutý na stav svojej sošky a odmieta sa presťahovať, ak pred novým domom stojí soška so stavom opačným od toho, na ktorý je zvyknutý zo svojho pôvodného domu. Alicka by chcela obyvateľom ulice pomôcť a poprepínať sošky tak, aby sa potom všetci naraz boli ochotní presťahovať.

Vždy, keď Alicka pôjde ulicou (či už na ceste do školy alebo späť), môže sa postupne pozerať na stavy holubov, okolo ktorých prechádza, a môže ich aj meniť (teda otvárať alebo zatvárať im krídla).

Alicka má plnú hlavu maturít, a tak si pri prechode ulicou **nepamätá nič o tom, čo videla a robila počas predchádzajúcich prechodov**. Našťastie má aspoň na papieri poznačený zoznam $a_0, a_1, \ldots, a_{N-1}$, takže vie, kto sa má kam sťahovať.

Nájdi pre Alicku stratégiu, pomocou ktorej dokáže zaručene dosiahnuť, že nastaví holuby tak, aby sa všetci boli ochotní sťahovať. Alicka sa môže ulicou prejsť najviac 60-krát. Na viac bodov však treba menej prechodov (viď sekciu Obmedzenia a hodnotenie).

Implementácia

Pri každom teste bude tvoj program postupne spustený viackrát.

Pri každom z týchto spustení dostaneš ten istý začiatok vstupu. V prvom riadku vstupu budú dve celé čísla w a N: poradové číslo prechodu ulicou a počet domov na nej. (Detaily o číslovaní prechodov si vysvetlíme nižšie.) V druhom riadku vstupu bude N celých čísel $a_0, a_1, ..., a_{N-1}$ udávajúcich, kto sa má kam sťahovať. Je zaručené, že čísla a_i tvoria *permutáciu* (teda obsahujú práve raz každú z hodnôt od 0 po $N-1$). Je možné, že pre niektoré i bude platiť $a_i = i$, teda niektorí obyvatelia môžu chcieť ostať vo svojich pôvodných domoch.

V rámci každého testu budú hodnoty N a a_i konštantné – teda pri každom spustení dostane tvoj program na vstupe tie isté hodnoty.

Beh programu s číslom nula

Pri prvom spustení dostane tvoj program na vstupe hodnotu $w = 0$. V tomto behu by mal na výstup vypísať jediné nezáporné celé číslo W ($0 \leq W \leq 60$) a skončiť. Vypísané číslo W udáva počet prechodov Alicky ulicou, ktoré chceš vykonať.

Tvoj program bude následne na tomto teste spustený ešte presne W-krát, teda raz pre každý prechod ulicou.

Nasledujúce behy programu

Pri prvom behu, ktorý simuluje prechod ulicou, dostane tvoj program na vstupe $w = 1$, pri druhom $w = 2$, a tak ďalej, až po posledný beh, v ktorom dostane $w = W$.

Po tom, ako zo vstupu prečítaš prvé dva riadky (s hodnotami w, N a a_i), začne prechod Alicky ulicou. Ten môže vyzerať dvoma spôsobmi:

- Ak je w nepárne, Alicka ide z domu do školy. Pri takomto prechode stretne domy v poradí $0, 1, ..., N-1$.

V takomto scenári má tvoj program fungovať nasledovne:

Najskôr načíta riadok s hodnotou b_0, udávajúcou aktuálny stav holuba pred domom 0. (Pripomíname, že hodnota 0 sú zavreté a 1 sú otvorené krídla.) Po načítaní riadku s b_0 by tvoj program mal vypísať jeden riadok s číslom 0 alebo 1: novým stavom krídel tohto holuba, teda hodnotou, na ktorú chceš nastaviť b_0.

Následne má tvoj program rovnako načítať riadok s aktuálnou hodnotou b_1, potom vypísať riadok s novou hodnotou b_1, potom načítať riadok s b_2, a tak ďalej až po koniec ulice. Po vypísaní novej hodnoty b_{N-1} musí tvoj program korektne skončiť.

Daj si pozor na to, že hodnotu b_{i+1} smieš načítať vždy až po tom, ako vypíšeš novú hodnotu b_i.

- Ak je naopak w párne, Alicka sa vracia domov, a teda ulicou prechádza v opačnom smere.

Tvoj program by mal fungovať rovnako ako pre nepárne w. Jediný rozdiel je v tom, že teraz najskôr načíta a vypíše b_{N-1}, potom b_{N-2}, a tak ďalej, až po b_0.

Pre $w = 1$ dostane tvoj program na vstupe hodnoty $b_0, b_1, ..., b_{N-1}$ zodpovedajúce začiatočným stavom holubov, teda tým, na ktoré sú jednotliví obyvatelia zvyknutí.

Pre každé $w > 1$ dostane tvoj program na vstupe tie hodnoty b_i, ktoré vypísal pri predchádzajúcom behu.

Po poslednom behu tvojho programu musí pre každé i platiť, že výsledná hodnota b_i je rovná pôvodnej hodnote b_{a_i}. Ak toto nebude platiť, dostaneš za príslušný test verdikt Wrong Answer.

Technické detaily

Testovač sčíta časy jednotlivých behov tvojho programu v rámci jedného testu. Ak tento súčet prekročí časový limit, dostaneš za tento test verdikt Time Limit Exceeded.

Tvoj program musí flushnúť štandardný výstup po vypísaní každého riadku výstupu. (Ak to nebudeš robiť, môžeš dostať verdikt Time Limit Exceeded.) V Pythone sa toto stane automaticky, ak budeš riadky vstupu čítať volaním `input()`. V C++ pri používaní streamov príkaz `cout << endl;` vypíše znak nového riadku a spraví flush. V C++ pri používaní `printf` na výpis treba potom zavolať samostatný príkaz `fflush(stdout)`.

Obmedzenia a hodnotenie

- $2 \leq N \leq 500$.
- Počet prechodov ulicou, ktoré tvoj program spraví, musí spĺňať $W \leq 60$.

Tvoje riešenie bude testované na viacerých sadách testov. V rámci každej sady všetky testy spĺňajú tie isté dodatočné obmedzenia. Ako obvykle, za sadu testov dostaneš body len vtedy, ak korektne vyriešiš všetky testy v nej.

Sady testov vyzerajú nasledovne:

Sada	Max bodov	Dodatočné obmedzenia
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 ďalších obmedzení

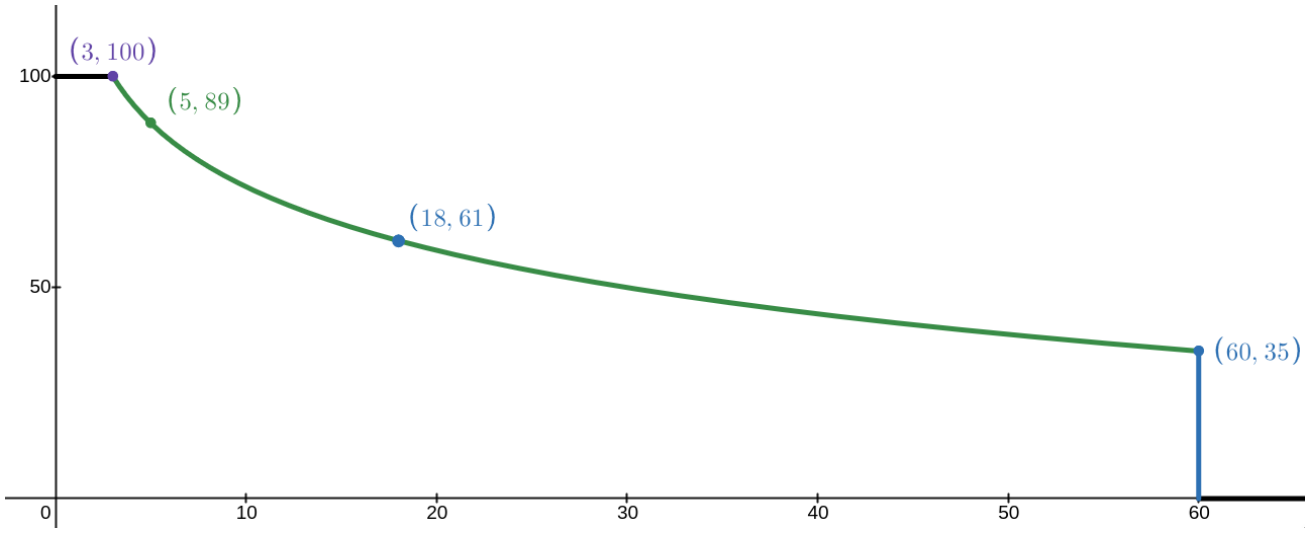
Za každú sadu, ktorej všetky testy korektne vyriešiš, budú tvoje body určené nasledovným vzorcom:

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

V tomto vzorci je S_g hodnota maximálneho počtu bodov za túto sadu a hodnota W_g je určená tak, že v každom teste z tejto sady sa pozrieme na hodnotu W, ktorú tvoj program potreboval, a z týchto hodnôt zoberieme maximum.

Výsledný počet bodov (za každú sadu vstupov zvlášť) je následne vždy zaokrúhlený na najbližšie celé číslo.

Graf na obrázku nižšie ukazuje, ako závisí tvoj počet bodov od najväčšej hodnoty W, ktorú tvoj program potrebuje. Konkrétne si všimni, že na plný počet bodov je potrebné mať v každom teste $W \leq 3$.



Pomocný nástroj na testovanie

Aby sa ti ľahšie lokálne testovalo tvoje riešenie, môžeš si stiahnuť jednoduchý nástroj (testing tool). V Kattise na stránke s touto úlohou ho nájdeš v sekcii “attachments”. Používanie tohto nástroja je dobrovoľné. Oficiálny grader použitý v testovači je úplne iný ako tento nástroj.

Nástroj použiješ nasledovne: Vytvor si vstupný súbor, napr. “sample1.in”. V ňom by mali byť tri riadky: v prvom riadku N, v druhom permutácia a_i a v treťom začiatočné hodnoty b_i. Teda napríklad takto:

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

Ak riešiš v Pythone a tvoje riešenie je `solution.py` (ktoré by si normálne spúšťala príkazom `pypy3 solution.py`), spusti toto:

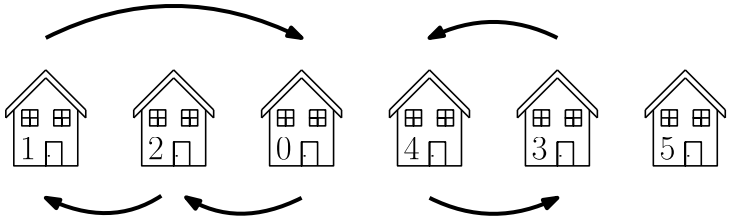
```
python3 testing_tool.py pypy3 solution.py < sample1.in
```

Ak riešiš v C++, najskôr svoje riešenie skompiluj (napr. príkazom `g++ -g -O2 -std=gnu++20 -static solution.cpp -o solution.out`) a potom spusti toto:

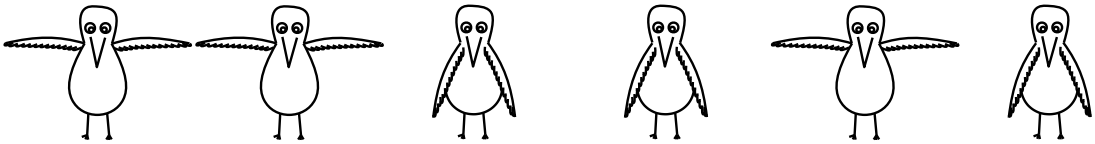
```
python3 testing_tool.py ./solution.out < sample1.in
```

Príklad

Majme nasledovnú permutáciu popisujúcu želané sťahovania:

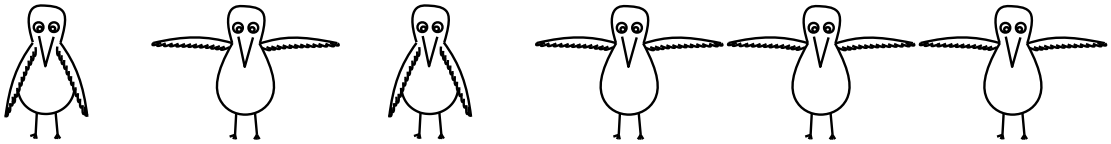


Pri spustení s $w = 0$ tvoj program vypíše $W = 2$, chce teda následne vykonať dva prechody Alicky ulicou. Pred jej prvým prechodom vyzerajú holuby v ulici nasledovne:



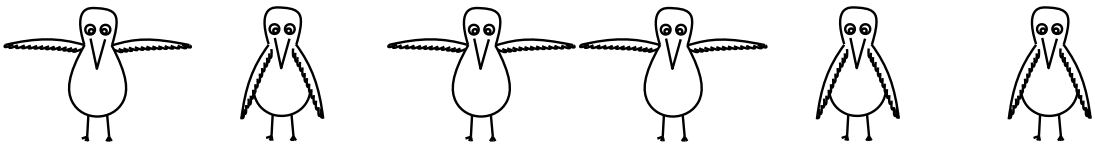
Teraz program spustíme s $w = 1$. Alicka postupne prejde ulicou zľava doprava a pomení stavy niektorých holubov. Pripomíname, že nový stav holuba i musí určiť skôr, ako sa pozrie na aktuálny stav holuba $i + 1$.

Po príchode Alicky do školy vyzerajú holuby v ulici nasledovne:



Poslednýkrát spustíme program, tentokrát s $w = 2$. Pri tomto spustení sa Alicka vracia zo školy domov. Postupne teda prechádza ulicou sprava doľava. Pri tomto prechode nový stav holuba i musí určiť skôr, ako sa pozrie na aktuálny stav holuba $i - 1$.

Po skončení tohto prechodu vyzerajú holuby nasledovne:



Ľahko overíme, že toto je správna konfigurácia holubov. Napríklad holub číslo 3 (teda štvrtý zľava, číslujeme od nuly) má otvorené krídla, čiže na konci máme $b_3 = 1$. Toto je správne, lebo do domu 3 sa má sťahovať človek 4 (máme $a_3 = 4$) a na začiatku mal holub pred domom 4 otvorené krídla (na začiatku platilo $b_4 = 1$).

Jednotlivé spustenia programu, ktoré zodpovedajú obrázkom vyššie, vyzerajú nasledovne

vstup tvojho programu	výstup tvojho programu
0 6	
1 2 0 4 3 5	
	2

vstup tvojho programu	výstup tvojho programu
1 6	
1 2 0 4 3 5	
1	
	0
1	
	1
0	
	0
0	
	1
1	
	1
0	
	1

vstup tvojho programu	výstup tvojho programu
2 6	
1 2 0 4 3 5	
1	
	0
1	
	0
1	
	1
0	
	1
1	
	0
0	
	1