D. Hagedekorasjoner

Problem Name	Garden Decorations
Tidsbegrensning	7 sekunder
Minnebegrensning	1 gigabyte

Hver dag på vei til og fra skolen går Detje langs en gate med N hus, nummerert fra 0 til $N-1$. Hus nummer i er for øyeblikket bebodd av person i. For å nyte nye omgivelser har beboerne i gaten bestemt seg for å bytte hus med hverandre. Personen som skal flytte inn i hus i er person a_i (som altså nå bor i huset med samme nummer, hus a_i).

Hvert hus har en fuglestatue i hagen. Statuene har to tilstander: Enten er vingene på fuglen *åpen* (som om fuglen flyr), eller er vingene *lukket* (som om fuglen står på bakken). Beboerne har veldig sterke meninger om hvordan fuglestatuen sin skal se ut. For øyeblikket er fuglen utenfor hus i satt i den tilstanden person i foretrekker. Beboerne nekter å flytte til et hus om ikke statuen foran det huset er satt til deres foretrukne tilstand. Detje ønsker å hjelpe dem å ordne fuglestatuene slik at beboerne kan gjennomføre planen sin om å flytte.

For å få til dette gjør hun følgende: Hver gang hun går langs gaten (enten på vei til eller fra skolen), observerer hun fuglene en etter en og gjør potensielt justeringer på statuene (ved å lukke eller åpne vingene). Siden dagene på skolen er travle, **husker hun ikke tilstanden til fuglene hun så på hennes forrige tur**. Heldigvis har hun skrevet ned listen $a_0, a_1, ..., a_{N-1}$, så hun vet hvilken beboer som skal flytte hvor.

Hjelp Detje å finne en strategi som forteller henne hvilke fugler som må manipuleres slik at hun kan justere fuglene til de foretrukne tilstandene. Hun kan gå langs gaten maksimalt 60 ganger, men en høyere score kan oppnås ved å gå færre ganger langs gaten.

Implementering

Dette er en *multi-run* oppgave, som betyr at programmet ditt vil bli kjørt flere ganger.

For hver gang programmet ditt kjøres, skal du først lese en linje med to heltall w og N, nummeret til turen du tar og antall hus. I første kjøring av programmet er $w = 0$, i den andre kjøringen er $w = 1$ og så videre (flere detaljer følger under).

På andre linje av input gis N heltall $a_0, a_1, ..., a_{N-1}$, som betyr at personen som skal flytte inn i hus i for tiden bor i hus a_i. Tallene a_i gir altså en *permutasjon* av tallene 0 til $N-1$: hvert av disse tallene opptrer nøyaktig én gang i listen $a_0, a_1, ..., a_{N-1}$. Merk at en beboer kan velge å ikke flytte; altså $a_i = i$ er en tillatt verdi.

Beboerene flytter bare en gang. Dette betyr at i en gitt test-case er verdien N og listen av tallene a_i det samme for hver gang programmet ditt kjører.

Første kjøring.

I den første kjøringen av programmet ditt er $w = 0$. I denne kjøringen skal du utelukkende skrive ut et heltall W ($0 \leq W \leq 60$), antallet ganger du ønsker at Detje skal gå forbi husene. Programmet ditt skal så avslutte. Etter dette vil programmet ditt bli kjørt W ganger.

Videre kjøringer.

I den neste kjøringen av programmet er $w = 1$. Etter dette vil $w = 2$ og så videre inntil den siste kjøringen hvor $w = W$.

Etter du har lest w, N og $a_0, a_1, ..., a_{N-1}$, begynner Detje å gå langs gaten.

- Hvis w er et oddetall går Detje til skolen, og vil passere husene i rekkefølge $0, 1, ..., N-1$.

  Programmet ditt skal nå lese en linje med b_0, som enten tar verdien 0 (lukket) eller 1 (åpen), den nåværende tilstanden til statuen foran hus 0. Etter å ha lest b_0 skal du skrive ut verdien du ønsker å sette b_0 til.

  Etter dette skal programmet ditt lese en linje med b_1, tilstanden til statuen foran hus 1; og så skrive ut den nye verdien til b_1. Dette fortsetter for hvert av de N husene. Etter Detje har passert det siste huset (altså når du leser og skriver b_{N-1}) **skal programmet ditt avslutte**.

  Merk at programmet ditt bare kan lese den neste verdien b_{i+1} etter at den har skrevet den nye verdien til b_i.

- Hvis w er et partall, går Detje hjem fra skolen, og vil derfor passere husene i omvendt rekkefølge, altså $N-1, N-2, ..., 0$. Altså, du starter med å lese og skrive b_{N-1}, så b_{N-2} og så videre helt til b_0.

Når $w = 1$ er input-verdiene $b_0, b_1, ..., b_{N-1}$ de originale tilstandene til statuene, og indikerer de foretrukne tilstandene til de respektive beboerene. Når $w > 1$ er input-verdiene $b_0, b_1, ..., b_{N-1}$ til programmet ditt det den forrige kjøringen av programmet ditt satt dem til.

På slutten, etter siste kjøring av programmet ditt, må verdien til b_i være satt til verdien foretrukket av personen som skal flytte inn i hus i, altså den originale verdien av b_{a_i} for alle i, ellers vil programmet ditt bedømmes *Wrong Answer*.

Detaljer.

Hvis *summen* av kjøretiden til de $W+1$ separate kjøringene av programmet ditt overstiger tidsbegrensningen vil besvarelsen din bedømmes som *Time Limit Exceeded*.

Forsikre deg om å flushe standard output etter å ha printet en linje, ellers kan programmet ditt bedømmes som *Time Limit Exceeded*. I Python blir dette gjort automatisk så lenge du benytter `input()` for å lese linjer. I C++ flusher `cout << endl;` i tillegg til å printe en ny linje. Anvender du printf, bruk `fflush(stdout)`.

Begrensninger og poenggiving

- $2 \leq N \leq 500$.
- Du kan benytte opp til $W \leq 60$ runder.

Løsningen din vil bli testet mot et sett testgrupper, hver verdt et visst antall poeng. Hver testgruppe inneholder en mengde tester. For å få poeng for en testgruppe må du løse alle testene i gruppen.

Gruppe	Max poeng	Begrensninger
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	Ingen ytterligere begrensninger

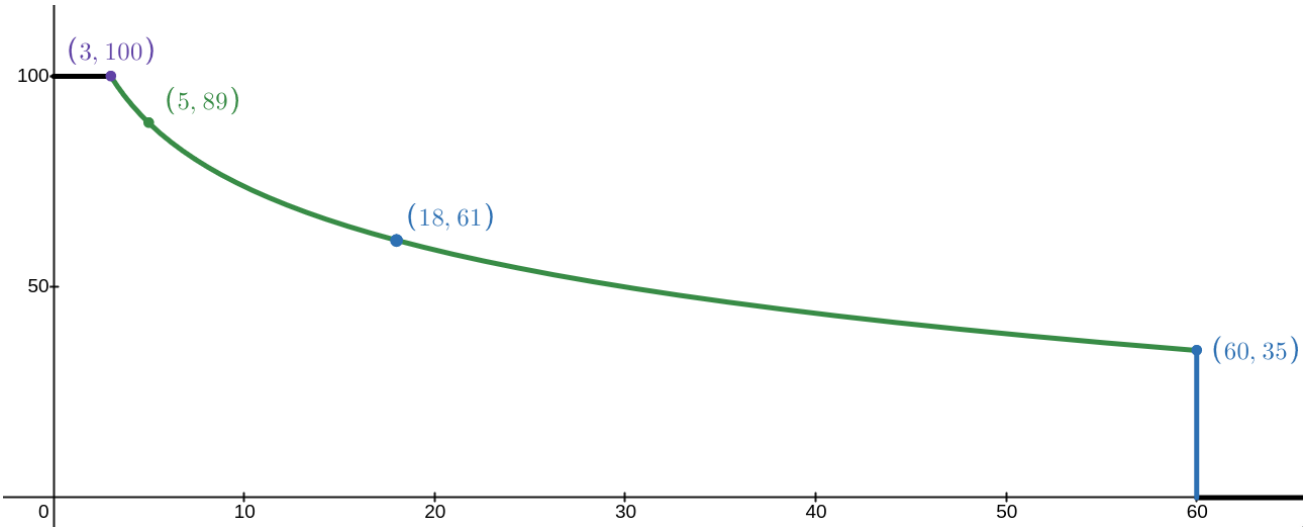
Hvor hver testgruppe som programmet ditt korrekt løser, får du en poengsum gitt ved følgende formel:

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

hvor S_g er maks poengscore for testgruppen og W_g er maksverdien til W brukt over alle testene i testgruppen.

Antall poeng for en testgruppe vil rundes opp til nærmeste heltall.

Grafen under viser antall poeng programmet ditt vil få, som en funksjon av W, hvor det antas at du løser alle testgruppene med samme verdi av W. For å få 100 poeng kreves for eksempel da at du løser hver testgruppe med $W \leq 3$.



Testverktøy

For å fasilitere testing av løsningen din, tilbyr vi et testverktøy som du kan laste ned. Se "attachments" i bunnen av oppgavens side på Kattis. Verktøyet er frivillig å anvende. Merk at den offisielle graderen på Kattis er ulik testverktøyet.

For å bruke testverktøyey, lag en input-fil, som for eksempel "sample1.in", som skal starte med et tall N, etterfulgt av en linje med N tall som spesifiserer permutasjonen, og en linje til med N bits (0 eller 1), som spesifiserer den opprinnelige tilstanden til fuglene. For eksempel:

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

For Pythonprogrammet, si `solution.py` (vanligvis kjørt som `pypy3 solution.py`):

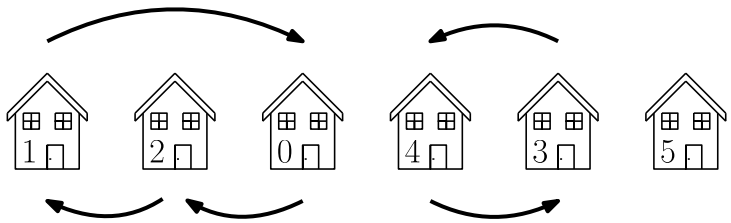
```
python3 testing_tool.py pypy3 solution.py < sample1.in
```

For C++-programmet, kompiler først med for eksempel `g++ -g -O2 -std=gnu++20 -static solution.cpp -o solution.out` og kjør så:

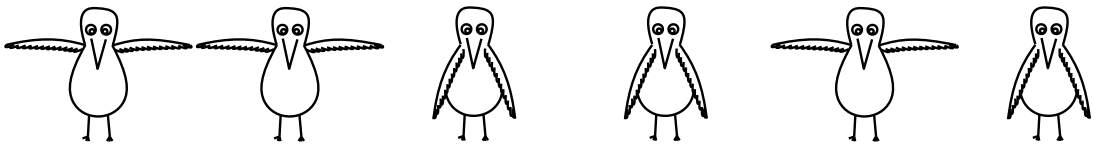
```
python3 testing_tool.py ./solution.out < sample1.in
```

Eksempel

I eksempelet er vi gitt følgende permutasjon av folk i husene:

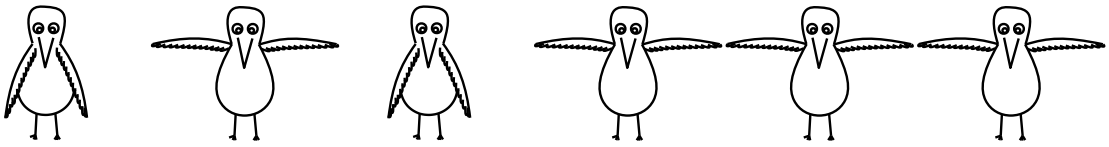


På første kjøring (med $w = 0$) skriver den ut $W = 2$, som betyr at Detje skal gå langs gaten to ganger (og programmet vil bli kjørt to ganger til). Før første tur, er fuglene i hagen i følgende tilstander:



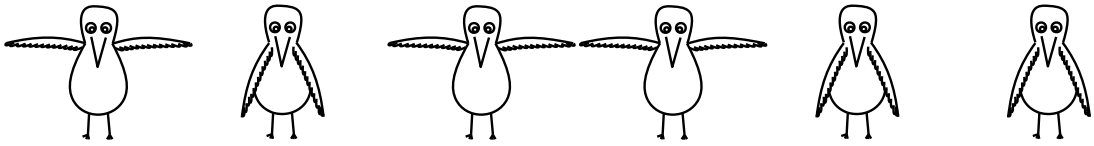
Programmet kjører så med $w = 1$, som indikerer Detjes første tur. Hun går forbi fuglene en etter en, fra venstre til høyre, og har anledning til å endre deres tilstander. Programmet må da skrive ut tilstanden for fugl nummer i før vi får se fugl nummer $i + 1$.

Etter Detje har ankommet skolen, er fuglene i følgende tilstander:



I siste kjøring av prorgrammet (med $w = 2$) går Detje hjem fra skolen. Husk at i dette tilfellet, går hun gjennom fuglene fra høyre til venstre! Det betyr at hun må bestemme tilstanden til fugl i før hun får se tilstanden til fugl nummer $(i - 1)$.

Etter turen er ferdig ser fuglene nå slik ut:



Dette er den riktige tilstanden. For eksempel er fuglestatue 3 (den fjerde fra venstre) åpen, så $b_3 = 1$. Dette er korrekt, ettersom person 4 skal flytte inn i hus 3 ($a_3 = 4$) og disse hadde først en åpen fuglestatue utenfor huset sitt (i utgangspunktet hadde vi $b_4 = 1$).

grader output	ditt output
0 6	
1 2 0 4 3 5	
	2

grader output	ditt output
1 6	
1 2 0 4 3 5	
1	
	0
1	
	1
0	
	0
0	
	1
1	
	1
0	
	1

grader output	ditt output
2 6	
1 2 0 4 3 5	
1	
	0
1	
	0
1	
	1
0	
	1
1	
	0
0	
	1