D. Flyttfåglar

Problemnamn	Flyttfåglar
Tidsgräns	7 sekunder
Minne	1 gigabyte

Varje dag påväg till och från skolan går Detje förbi en gata med N hus som är numrerade från 0 till $N-1$. Just nu bor person i i hus nummer i. De har dock tröttnat på sina nuvarande hus och har därför bestämt sig för att byta hus med varandra. Mer konkret betyder det att person a_i (som för tillfället bor i hus a_i) kommer att flytta in i hus i.

Varje hus har en fågelstaty i trädgården. Statyerna kan antingen ha utfällda vingar eller hopfällda vingar. Invånarna har väldigt starka preferenser angående fåglarnas utseende och vägrar därför att flytta till ett hus där fågeln inte ser likadan ut som i den förra trädgården. Detje vill hjälpa dem genom att ändra fåglarnas utseende på ett sådant sätt så att alla personer kan flytta in i sina nya hus.

För att göra det behöver hon följande: när hon går längs med gatan (antingen påväg till eller från skolan) så betraktar hon fåglarna hon passerar en i taget och justerar potentiellt statyerna (genom att antingen öppna eller stänga vingarna). Eftersom både skolan och hennes fritid är väldigt intensiv så **kommer hon inte ihåg hur fåglarna såg ut under någon tidigare promenad**. Som tur är har hon skrivit ned listan $a_0, a_1, ..., a_{N-1}$, så att hon kommer ihåg vilken person som flyttar var.

Hjälp Detje att designa en strategi som beskriver hur hon ska ändra på fåglarna så att personerna kan flytta in som önskat. Hon kan gå längs med gatan maximalt 60 gånger, men för att få mer poäng krävs det att hon går färre gånger.

Implementation

Ditt program kommer att köras flera gånger.

I varje körning läser du först in en rad med de två talen w och N, indexet som promenaden har samt antalet hus som finns. I första körningen av ditt program är $w = 0$, i den andra är $w = 1$, och så vidare (mer detaljer kring detta finns nedan).

På den andra raden finns det N heltal $a_0, a_1, ..., a_{N-1}$, vilket betyder att personen som vill flytta in i hus i för tillfället bor i hus a_i. Listan av alla a_i:n bildar en *permutation*: det vill säga, varje tal från 0 till $N-1$ förekommer exakt en gång i listan. Notera att en person inte nödvändigtvis behöver flytta, vilket innebär att $a_i = i$ är tillåtet.

Personerna kan endast flytta en gång. Det innebär att för ett givet testfall kommer N och listan av a_i:n vara samma varje gång ditt program körs.

Första körningen.

Under första körningen av ditt programmet är $w = 0$.

Under denna körning ska du enbart skriva ut ett heltal W ($0 \leq W \leq 60$) – antalet gånger som du vill att Detje ska gå längs med gatan. Ditt program ska sedan avsluta. Efter det kommer ditt program att köras W gånger till.

Efterföljande körningar.

Under nästkommande körning av programmet är $w = 1$, i den efteråt är $w = 2$, och så vidare till och med sista körningen då $w = W$.

Efter att du har läst in w, N och $a_0, a_1, ..., a_{N-1}$, kommer Detje att börja gå längs med gatan.

- om w är ett udda tal är Detje påväg till skolan, och hon kommer passera husen i ordningen $0, 1, ..., N-1$.

  Ditt program ska nu läsa in en rad som innehåller b_0, antingen 0 (hopfällda vingar) eller 1 (utfällda vingar), vilket beskriver hur statyn utanför hus 0 ser ut. Efter att du har läst in b_0 ska du skriva ut en rad som antingen består av 0 eller 1, vilket beskriver hur statyn kommer se ut efter promenaden.

  Efter det ska programmet läsa in en rad med b_1, hur statyn utanför hus 1 ser ut; och skriva ut hur statyn kommer se ut efter promenaden.

  Samma procedur upprepas sedan för resterande hus. Efter att du har passerat det sista huset (det vill säga att du läser in och skriver ut ett värde för b_{N-1}) **ska ditt program avsluta.**

  Notera att ditt program enbart kan läsa värdet b_{i+1} efter att du har skrivit ut värdet för b_i.

- Om w är ett jämnt tal är Detje påväg från skolan, och hon kommer att passera husen i den omvända ordningen $N-1, N-2, ...0$.

  Processen för att ändra fåglarna är densamma som när w är ett udda tal, förutom att du börjar med att läsa in och skriva ut b_{N-1}, sedan b_{N-2}, och så vidare, ända till b_0.

När $w = 1$ beskriver indatavärdena $b_0, b_1, ..., b_{N-1}$ hur fågelstatyerna såg ut från början. När $w > 1$ beskriver indatavärdena $b_0, b_1, ..., b_{N-1}$ hur fågelstatyerna såg ut efter din föregående körning.

Slutligen gäller det att efter att ditt program har kört för sista gången så måste värdet av b_i vara lika med det ursprungliga värdet av b_{a_i} för varje i, annars kommer din inskickning att dömas som Wrong Answer.

Detaljer.

Om *summan* av körtiderna av de $W + 1$ olika körningarna av ditt program överskrider tidsgränsen så kommer din inskickning att dömas som Time Limit Exceeded.

Säkerställ att du flushar utdatan efter att du skriver ut varje rad, annars kan din inskickning dömas som Time Limit Exceeded. Python flushar automatiskt om du använder `input()` för att läsa indatan. I C++ kommer `cout<<endl;` att flusha (samt skriva en newline); om du istället använder printf, skriv `fflush(stdout)` efteråt.

Begränsningar och poänggrupper

- $2 \leq N \leq 500$.
- Du får använda max $W \leq 60$ promenader.

Din lösning kommer att testas på flera testgrupper, som var och en är värda ett antal poäng. Varje testgrupp innehåller flera testfall. För att få poängen för en testgrupp måste du lösa alla testfall i testgruppen.

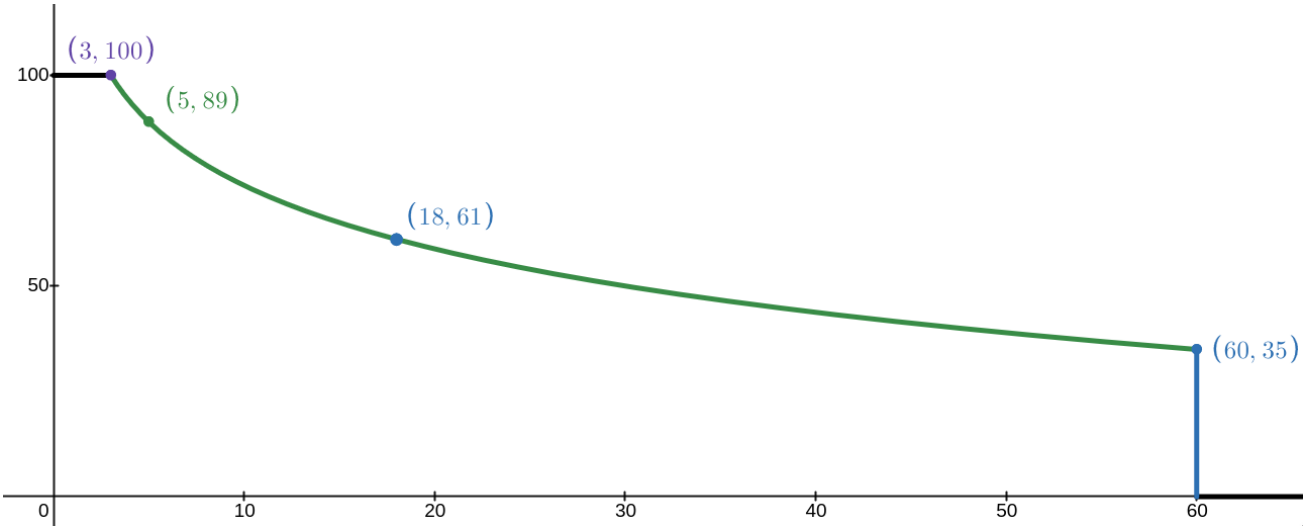
Grupp	Maxpoäng	Begränsningar
1	10	$N = 2$
2	24	$N \leq 15$
3	9	$a_i = N - 1 - i$
4	13	$a_i = (i + 1) \bmod N$
5	13	$a_i = (i - 1) \bmod N$
6	31	Inga ytterligare begränsningar

För varje testgrupp som ditt program löser kommer du att få poäng enligt följande formel:

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

där S_g är den maximala poängen för en testgrupp, och W_g är det maximala värdet av W som används för något testfall i testgruppen. Din poäng för en testgrupp avrundas till närmsta heltal.

Grafen nedan visar antalet poäng som en funktion av W som ditt program kommer att få om det löser alla testgrupper med samma värde av W. I synnerhet gäller det att för att få 100 poäng på problemet så behöver varje testfall lösas med $W \leq 3$.



Testverktyg

För att underlätta testningen av din lösning tillhandahåller vi ett enkelt verktyg som du kan ladda ned. Se "attachments" längst ned på problemsidan i Kattis. Det är valfritt att använda verktyget. Notera att det officiella domarprogrammet på Kattis är annorlunda jämfört med testverktyget.

För att använda verktyget behöver du skapa en indatafil, exempelvis "sample1.in". Det ska först innehålla en rad med talet N, sedan en till rad med N tal som är permutationen, och slutligen en till rad med N bitar (0:or och 1:or) som är fåglarnas ursprungliga positioner. Exempelvis:

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

Ett Pythonprogram, exempelvis `solution.py`(som normalt körs med `pypy3 solution.py`) körs nu med:

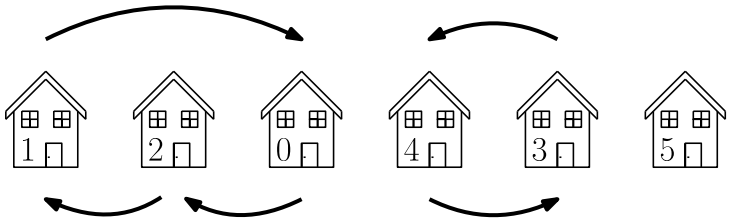
```
python3 testing_tool.py pypy3 solution.py < sample1.in
```

Ett C++-program ska först kompileras (exempelvis med `g++ -g -O2 -std=gnu++20 -static solution.cpp -o solution.out`), och sedan köras med:

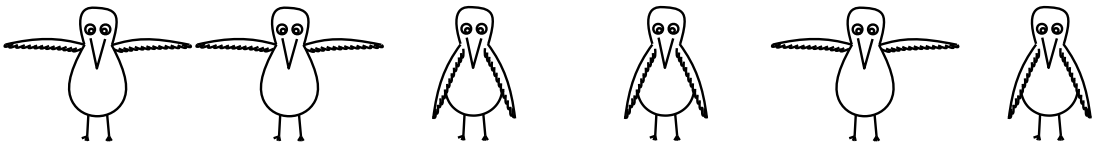
```
python3 testing_tool.py ./solution.out < sample1.in
```

Exempelfall

I exempelfallet ges vi först följande permutation av invånarna på gatan:

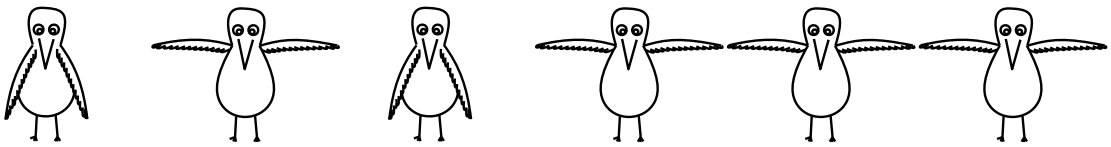


Första gången som programmet körs (med $w = 0$), skrivs $W = 2$ ut, vilket innebär att Detje kommer att gå längs med gatan två gånger (och att programmet kommer att köras två ytterligare gånger). Innan första promenaden ser fåglarna ut såhär:



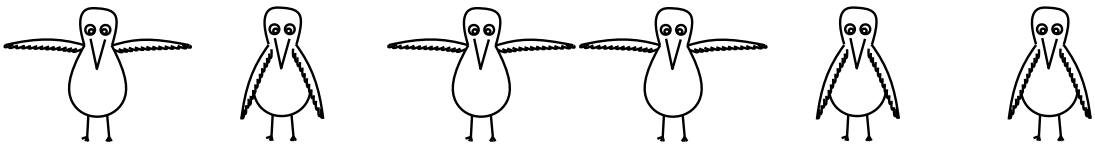
Sedan körs programmet med $w = 1$: Detjes första promenad. Hon passerar varje fågel från vänster till höger och ändrar potentiellt en del av deras utseenden. Exempelprogrammet måste skriva ut hur den i:te fågeln ser ut innan det ser den $(i + 1)$:te fågeln.

Efter att Detje kommer fram till skolan ser fåglarna ut såhär:



I den sista körningen av programmet (där $w = 2$) är Detje påväg hem från skolan. Kom ihåg att hon därför kommer att gå igenom fåglarna från höger till vänster, alltså i omvänd ordning! Det innebär att hon måste bestämma hur den i:te fågeln ska se ut innan hon ser den $(i - 1)$:te fågeln.

Efter att Detje kommer hem från skolan ser fåglarna ut såhär:



Det är det korrekta utseendet för alla fåglarna. Exempelvis har fågelstaty 3 (det vill säga den fjärde från vänster) utfällda vingar ($b_3 = 1$), vilket stämmer då person 4 kommer att flytta in där ($a_3 = 4$) och hon hade från början en staty med utfällda vingar (ursprungligen var $b_4 = 1$).

graderns utdata	din utdata
0 6	
1 2 0 4 3 5	
	2

graderns utdata	din utdata
1 6	
1 2 0 4 3 5	
1	
	0
1	
	1
0	
	0
0	
	1
1	
	1
0	
	1

graderns utdata	din utdata
2 6	
1 2 0 4 3 5	
1	
	0
1	
	0
1	
	1
0	
	1
1	
	0
0	
	1