D. Bahçe Dekorasyonları

Problem Adı	Bahçe Dekorasyonları
Zaman Limiti	7 saniye
Bellek Limiti	1 gigabyte

Her gün okula giderken ve eve dönerken Detje, 0'dan $N-1$'e kadar numaralandırılmış N evin bulunduğu bir sokakta yürür. i numaralı evde i kişisi oturmaktadır. Manzarayı değiştirmek için, ev sakinleri birbirleriyle ev değiştirmeye karar verdiler. Ev değişiklikleri sonucunda, i numaralı eve taşınacak kişi, a_i kişisidir (şu anda a_i evinde yaşayan kişidir).

Her evin bahçesinde bir kuş heykeli bulunmaktadır. Heykellerin iki olası durumu vardır: ya kanatları *açık* (kuş uçuyormuş gibi) ya da *kapalı* (kuş yerde duruyormuş gibi) olabilir. Sakinler, kuş heykellerinin nasıl görünmesi gerektiği konusunda çok kesin tercihlere sahiptirler. Şu anda, ev i'nin önündeki kuş heykeli, i sakininin tercih ettiği durumdadır. Sakinler, heykel tercih ettikleri duruma ayarlanmadıkça bir eve taşınmayı reddederler. Detje, onların taşınmalarına yardımcı olmak için kuş heykellerini düzenlemek istiyor.

Bunun için şöyle yapar: sokakta yürüdüğünde (okula giderken veya eve dönerken), karşılaştığı kuşları birer birer gözlemler ve bazı heykelleri kanatlarını açarak veya kapatarak) ayarlayabilir. Okulda ve evde günleri çok yoğun geçtiği için, **önceki yürüyüşlerinde gördüğü kuşların durumunu hatırlamaz.** Neyse ki, $a_0, a_1, ..., a_{N-1}$ listesini yazmış, böylece hangi sakinin nereye taşındığını biliyor.

Detje'ye, heykelleri sakinlerin isteklerine göre ayarlamak için hangi kuşları ayarlaması gerektiğini söyleyen bir strateji tasarlamasında yardımcı olun. En fazla 60 kez sokak boyunca yürüyebilir, ancak daha yüksek bir puan elde etmek için daha az yürüyüş yapmalıdır.

Gerçekleştirme

Bu, programınızın birden fazla kez çalıştırılacağı bir çoklu çalıştırma (multirun) problemidir.

Her çalıştırmada, ilk olarak iki tam sayı (w ve N) içeren bir satır okuyacaksınız; bu sayılar sırasıyla yürümenin indeksi (index of the walk) ve evlerin sayısını belirtir. Programınızın ilk çalıştırmasında $w = 0$, ikinci çalıştırmasında $w = 1$, ve bu şekilde devam eder (detaylar aşağıda açıklanmıştır).

Girdinin ikinci satırında N tane tam sayı ($a_0, a_1, ..., a_{N-1}$) bulunmaktadır. Bu, ev i'ye taşınacak kişinin şu anda ev a_i'de yaşadığı anlamına gelir. a_i değerleri bir *permütasyon* oluşturur: yani, 0'dan $N-1$'e kadar her sayı a_i listesinin içinde tam olarak bir kez görülür. Bir sakinin taşınmamayı seçebileceğini unutmayın; yani, $a_i = i$ olması kabul edilebilir.

Sakinler sadece bir kez ev değiştirir. Bu, sabit bir test durumu için N değeri ve a_i listesinin programınızın tüm çalıştırmalarında aynı kalacağı anlamına gelir.

İlk Çalıştırma.

Programınızın ilk çalıştırmasında, $w = 0$'dır. Bu çalıştırmada, sadece tek bir tam sayı W ($0 \leq W \leq 60$) çıktısı vermelisiniz; bu, Detje'nin evlerin önünden kaç kez geçmesini istediğinizi belirten sayıdır. Ardından programınız çıkış yapmalıdır. Bunu takiben, programınız W kez daha çalıştırılacaktır.

Takip Eden Çalıştırmalar.

Programınızın bir sonraki çalıştırmasında, $w = 1$; ondan sonrakinde $w = 2$; ve bu şekilde son yürütmeye kadar $w = W$ olacaktır.

w, N ve $a_0, a_1, ..., a_{N-1}$ değerlerini okuduktan sonra, Detje sokak boyunca yürümeye başlar.

- Eğer w bir tek sayı ise, Detje evinden okula yürür ve evlerin önünden $0, 1, ..., N-1$ sırasıyla geçer.

  Programınız şimdi ev 0'ın önündeki heykelin mevcut durumu olan b_0'ı içeren satırı okuyacaktır, bu durum ya 0 (kapalı) ya da 1 (açık) olabilir. b_0'ı okuduktan sonra, b_0'ı ayarlamak istediğiniz yeni değeri (0 veya 1) içeren bir çıktı vermelisiniz.

  Sonra programınız ev 1'in önündeki heykelin durumu olan b_1'i okuyacak ve yeni b_1 değerinin çıktısını verecektir. Bu, her bir N ev için devam eder. Son evi (yani b_{N-1}'i okuduktan ve yazdırdıktan sonra) geçtikten sonra **programınız çıkış yapmalıdır.**

  Programınızın yalnızca b_i değerini yazdırdıktan sonra b_{i+1} değerini okuyabileceğini unutmayın.

- Eğer w bir çift sayı ise, Detje okuldan evine yürür ve bu sefer evlerin önünden ters sırayla $N-1, N-2, ..., 0$ geçer. Yani, ilk olarak b_{N-1}'i okuyup yazdırarak başlayıp, ardından b_{N-2}, ve bu şekilde b_0'a kadar devam etmelisiniz.

$w = 1$ olduğunda, girdi değerleri $b_0, b_1, ..., b_{N-1}$ kuş heykellerinin orijinal durumudur. $w > 1$ olduğunda, programınıza gelen $b_0, b_1, ..., b_{N-1}$ girdi değerleri, programınızın bir önceki çalıştırılmasında ayarladığı değerler olacaktır.

Son çalıştırmadan sonra, b_i değerinin tüm i'ler için b_{a_i}'nin orijinal değerine eşit olması gerekir, aksi takdirde "Wrong Answer" sonucu alırsınız.

Detaylar.

Eğer programınızın $W+1$ ayrı çalıştırmasının *toplam* çalışma süresi zaman limitini aşarsa, gönderiminiz "Time Limit Exceeded" olarak değerlendirilecektir.

Her satırı yazdırdıktan sonra standart çıktıyı temizlediğinizden emin olun, aksi takdirde programınız "Time Limit Exceeded" olarak değerlendirilebilir. Python'da, input() kullanarak satırları okuduğunuz sürece bu otomatik olarak gerçekleşir. C++'da, cout << endl; bir satır sonu yazdırmanın yanı sıra tamponu (flushes the buffer) temizler; eğer printf kullanıyorsanız, fflush(stdout) kullanın.

Kısıtlar ve Puanlama

- $2 \leq N \leq 500$.
- $W \leq 60$.

Çözümünüz, belirli puan değerine sahip bir dizi test grubunda test edilecektir. Her test grubu bir dizi test durumu içerir. Bir test grubunun puanlarını alabilmek için, o test grubundaki tüm test durumlarını çözmeniz gerekmektedir.

Grup	Maksimum puan	Limitler
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	Ek kısıt yoktur.

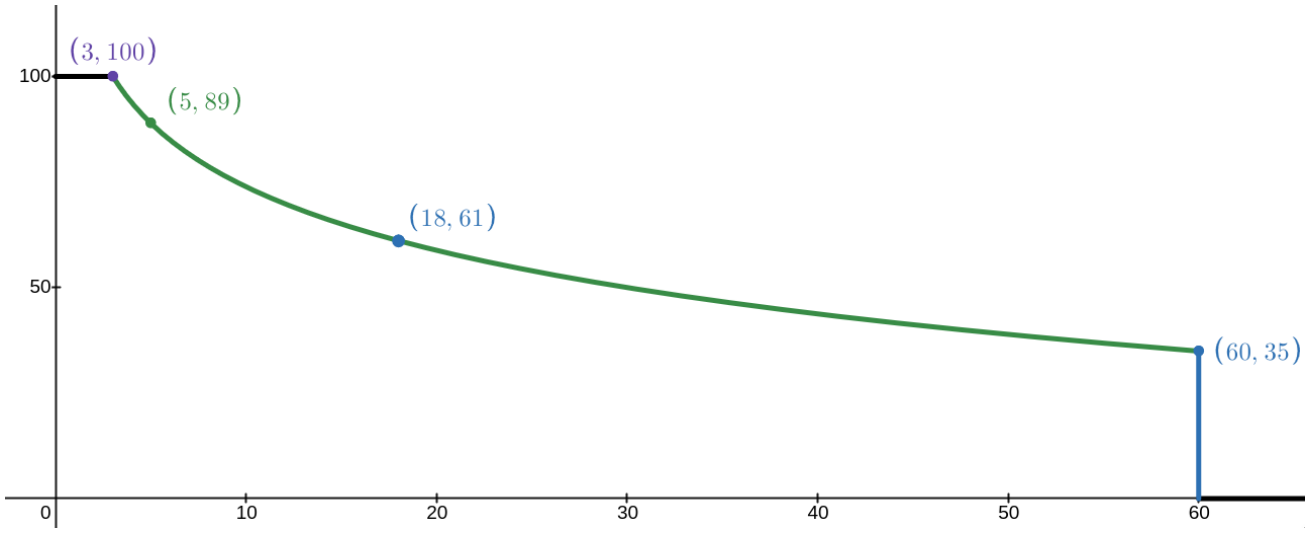
Programınızın doğru çözdüğü her test grubu için, aşağıdaki formüle dayalı olarak bir puan alacaksınız:

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

Burada S_g test grubu için maksimum puanı, ve W_g test grubundaki herhangi bir test durumu için kullanılan maksimum W değerini gösterir. Her test grubu için puanınız en yakın tam sayıya yuvarlanacaktır.

Aşağıdaki grafik, programınızın tüm test gruplarını aynı W değeri ile çözmesi durumunda alacağı puanları W'nin bir fonksiyonu olarak göstermektedir. Özellikle, bu problemde 100 puan elde

etmek için her test vakasını $W \leq 3$ ile çözmeniz gerekmektedir.



Test Aracı

Çözümünüzü test etmenizi kolaylaştırmak için indirip kullanabileceğiniz basit bir araç sağlıyoruz. Kattis problem sayfasının altındaki "ekler" bölümüne bakın. Bu aracın kullanımı isteğe bağlıdır. Resmi grader programının Kattis üzerindeki test aracından farklı olduğunu unutmayın.

Aracı kullanmak için, "sample1.in" gibi bir girdi dosyası oluşturun. Bu dosya, bir tam sayı (N) ile başlamalı, ardından permütasyonu belirten N adet sayı içeren bir satır ile devam etmeli ve kuşların başlangıç durumlarını belirten N adet bit (0 veya 1) içeren başka bir satırla devam etmelidir. Örneğin:

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

Python programları için, diyelim ki `solution.py` (normalde `pypy3 solution.py` olarak çalıştırılır):

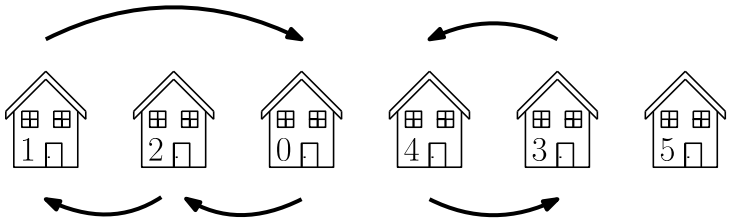
```
python3 testing_tool.py pypy3 solution.py < sample1.in
```

C++ programları için, önce derleyip: (örneğin `g++ -g -O2 -std=gnu++20 -static solution.cpp -o solution.out`) sonra çalıştırın:

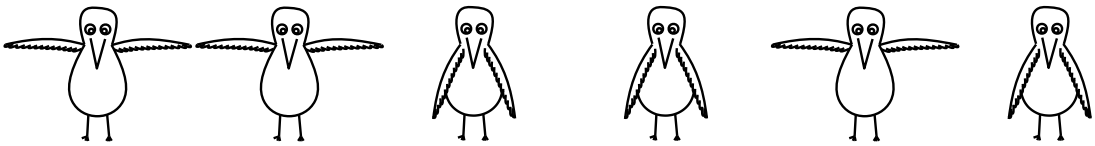
```
python3 testing_tool.py ./solution.out < sample1.in
```

Örnek

Örnekte, evlerde yaşayan kişilerin aşağıdaki permütasyonu verilmiştir:

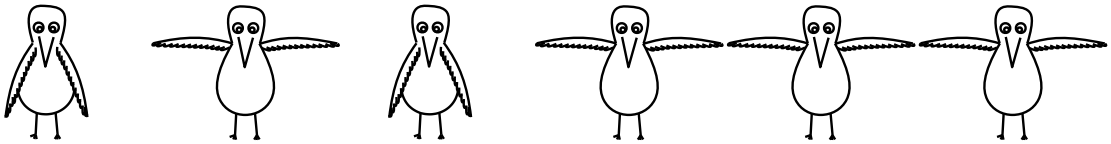


Örnek program ilk çalıştırıldığında ($w = 0$ ile), $W = 2$ çıktısı verir, bu da Detje'nin sokak boyunca iki kez yürüyeceği (ve programın iki kez daha çalıştırılacağı) anlamına gelir. İlk yürüyüşten önce, bahçelerdeki kuşlar şu şekildedir:



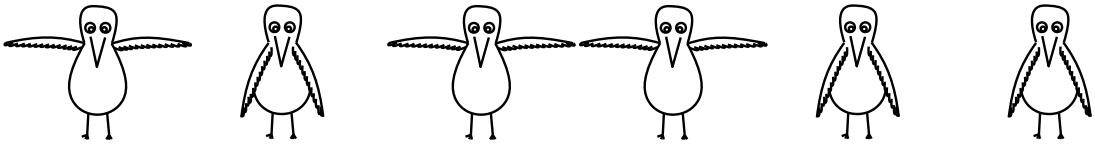
Daha sonra program $w = 1$ ile çalıştırılır: Bu, Detje'nin ilk yürüyüşünü (evden okula) gösterir. Detje soldan başlayarak kuşların yanından birer birer geçer, ve bir ihtimal (possibly) onların durumlarını değiştirir. Örnek program, i'inci kuşun durumunu, $i + 1$'inci kuşu görmeden önce çıktı olarak vermelidir.

Detje okula vardıktan sonra, kuşların durumu şu şekildedir.



Programın son çalıştırılmasında ($w = 2$ ile), Detje okuldan eve yürür. Bu durumda, kuşların yanından sağdan sola geçerek ve bu sırada işleyerek ilerler! Bu, i'inci kuşun durumunu $i - 1$'inci kuşu görmeden önce belirlemesi gerektiği anlamına gelir.

Yürüyüşünü bitirdikten sonra, kuşlar şimdi şöyle görünür:



Gerçekten de, bu doğru konfigürasyondur. Örneğin, kuş heykeli 3 (yani soldan dördüncü) açık (şu anda $b_3 = 1$), bu doğrudur çünkü kişi 4 oraya taşınacaktır ($a_3 = 4$) ve o başlangıçta açık bir kuş heykeline sahipti (başlangıçta $b_4 = 1$).

grader çıktısı	sizin çıktınız
0 6	
1 2 0 4 3 5	
	2

grader çıktısı	sizin çıktınız
1 6	
1 2 0 4 3 5	
1	
	0
1	
	1
0	
	0
0	
	1
1	
	1
0	
	1

grader çıktısı	sizin çıktınız
2 6	
1 2 0 4 3 5	
1	
	0
1	
	0
1	
	1
0	
	1
1	
	0
0	
	1