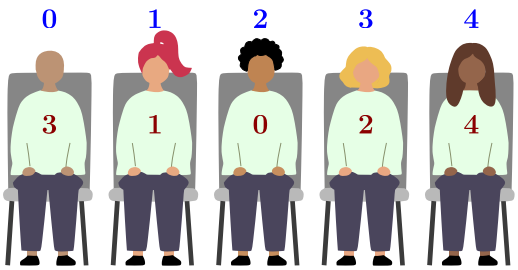
D. Oturma Planı (seatingplan)

Bu EGOI'nin kapanış törenine N önemli konuk katılacak. Bunların hepsi, diplomatik protokolün tüm inceliklerine uyan çok belirli bir sırada ön sıraya oturtulmalıdır. Doğru oturma sırasını belirlemek Noemi'nin iki uykusuz gecesine mal oldu.

Kapanış töreninden Veronica sorumlu. Onun pek çok sorumluluğundan biri de ön sıradaki koltuklarda doğru isim tabelalarının olduğundan emin olmak. Tek bir küçük sorun var: Noemi ona doğru oturma sırasını hiç söylemedi ve şimdi Noemi ortalarda yok. Neyse ki fotoğrafçı Dorka'nın işine yarayabilecek bir uygulaması var.

Dorka, ön sıradaki konukların belirli fotoğraflarını çekebilmek için kameralarını hazırlamalıydı. Kurulum için her fotoğrafın ne kadar geniş olacağını bilmesi gerekiyordu, bu yüzden Noemi onun için ihtiyacı olan bilgiyi hızlıca veren bir uygulama yaptı. Veronica şimdi doğru koltuk atamasını bulmak için bu uygulamayı kullanmak istiyor.

N önemli konuk 0'dan $N-1$'e kadar numaralandırılmıştır. Ön sıradaki koltuklar da soldan sağa doğru 0'dan $N-1$'e kadar numaralandırılmıştır. Her bir I $(0 \leq I \leq N-1)$ için, g_I, I numaralı koltukta oturması gereken konuğu; s_I ise I numaralı konuğun oturması gereken koltuğu temsil etsin.



Şekil 1: Beş konuklu bir sıra. Bu sıra için $g = [3, 1, 0, 2, 4]$ ve $s = [2, 1, 3, 0, 4]$.

Uygulama şu şekilde çalışır:

- Dorka, tam olarak üç farklı konuğun I, J, K numaralarını girer.
- Uygulama, seçilen üç konuğun da fotoğrafta olması durumunda görülebilecek minimum konuk sayısını ona söyler. Resmi olarak, uygulama $\max(s_I, s_J, s_K) - \min(s_I, s_J, s_K) + 1$ değerini görüntüler.

Örneğin, Şekil 1 kısmında gösterilen duruma bakın:

- $I = 0$, $J = 2$ ve $K = 4$ konukları $s_I = 2$, $s_J = 3$ ve $s_K = 4$ numaralı koltuklardadır. Eğer Dorka onları seçerse, uygulama $\max(2, 3, 4) - \min(2, 3, 4) + 1 = 3$ değerini görüntüler.

  Başka bir deyişle, 0, 2 ve 4 numaralı konukları içeren en dar fotoğraf sadece bu üç konuğu içerir.

- $I = 0$, $J = 4$ ve $K = 3$ konukları $s_I = 2$, $s_J = 4$ ve $s_K = 0$ numaralı koltuklardadır. Eğer Dorka onları seçerse, uygulama $\max(2, 4, 0) - \min(2, 4, 0) + 1 = 5$ değerini görüntüler.

  Başka bir deyişle, verilen üç konuğu içeren bir fotoğraf, 5 konuğun hepsini içermelidir.

Veronica'nın doğru oturma sırasını Dorka'nın uygulamasını kullanarak belirlemesine yardım edin. Daha spesifik olarak, programınız $g_0, g_1, ..., g_{N-1}$ dizisini belirlemeli ve çıktı olarak vermelidir. Her

zaman tam olarak iki doğru cevap vardır (biri diğerinin tersidir) ve ikisinden birini çıktı olarak verebilirsiniz. Puanınız, çözümünüzün uygulamaya yaptığı sorgu sayısına bağlı olacaktır.

Implementasyon

> ⇨ Bu etkileşimli bir problemdir. Programınız, aşağıda açıklanan formatta bir değerlendirici ile iletişim kurmak için standart girdi ve çıktıyı kullanacaktır.

Programınız, takip eden test durumu sayısını belirten bir pozitif tam sayı T içeren bir girdi satırını okuyarak başlamalıdır.

Her test durumu için programınız, koltuk sayısı (aynı zamanda konuk sayısı) olan pozitif bir tam sayı N içeren bir girdi satırını okuyarak başlamalıdır.

Bir sorgu yapmak için, programınız "? I J K" formatında bir satır çıktısı vermelidir; burada $0 \leq I, J, K \leq N-1$ üç **farklı** sayıdır.

Bir sorgu yaptıktan sonra programınız, sorgunuzun cevabı olan bir pozitif tam sayı içeren bir satır okumalıdır.

Doğru oturma sırasını cevaplamak için programınız "! g_0 … g_{N-1}" formatında bir satır çıktısı vermelidir.

Tüm T test durumlarını çözdükten sonra programınız normal şekilde sonlanmalıdır.

CMS'de çözümünüzü test etmek için kullanılan resmi değerlendiricinin **adaptif** (uyarlanabilir) olabileceğini unutmayın. Yani, bazı test durumları için konukların permütasyonu önceden belirlenmemiştir. Bunun yerine değerlendirici, programınız tarafından halihazırda sorulan sorgulara bağlı olarak kalan permütasyonlardan hangisini kullanacağına karar verebilir.

Flush İşlemi. Sağlanan şablonları kullanmıyorsanız, her satırı yazdırdıktan sonra standart çıktıyı temizlediğinizden (flush) emin olun, aksi takdirde programınız *Yanlış* olarak değerlendirilebilir. Python'da, satırları okumak için `input()` kullanırsanız bu otomatik olarak gerçekleşir ve temizlemeye zorlamak için `print(..., flush=True)` kullanabilirsiniz. C++'ta, `cout << endl;` yeni bir satır yazdırmanın yanı sıra temizleme işlemi de yapar; printf kullanıyorsanız `fflush(stdout)` kullanın.

Kısıtlamalar

- $1 \leq T \leq 10$.
- N değeri 5 (sadece örnek), 8, 40 veya 2000 olacaktır.
- Her test durumu için en fazla 10 000 sorgu yapabilirsiniz.

Puanlama

Çözümünüz birkaç alt göreve ayrılmış test durumlarında test edilecektir. Bir alt görevden puan alabilmek için, o görevdeki tüm testleri doğru şekilde çözmeniz gerekir.

- **Alt görev 0 [0 puan]**: Örnek ($N = 5$).
- **Alt görev 1 [9 puan]**: $N = 8$.
- **Alt görev 2 [11 puan]**: $N = 2000$ ve 0 ile 1 numaralı konuklar yan yana oturuyor.
- **Alt görev 3 [15 puan]**: $N = 40$.
- **Alt görev 4 [65 puan]**: $N = 2000$.

1 ve 2. alt görevler için, tüm test durumlarını doğru şekilde çözen herhangi bir çözüm tüm puanları alacaktır.

3 ve 4. alt görevler için çözümünüzün puan alabilmesi için tüm test durumlarını doğru şekilde çözmesi gerekir ve puanınız, bir test durumunu çözmek için yapmanız gereken en fazla sorgu sayısı olan Q_s'ye bağlı olacaktır. $X_s = \max(1, Q_s/N)$ olsun. 3 ve 4. alt görevlerin puanları şu şekilde hesaplanır:

$$\text{score}_3 = \min\left(15, 3 + \frac{19}{X_s^{1.5}}\right), \qquad \text{score}_4 = \min\left(65, 3 + \frac{91}{X_s^{1.5}}\right)$$

score_s değeri alt görev başına en yakın tam sayıya yuvarlanır ve toplam puanınız bunların toplamıdır. Tam puan almak için 3. alt görevi en fazla 55 sorguda ve 4. alt görevi en fazla 2597 sorguda çözmeniz gerekir. Q_s'nin örnek değerleri ve 3 ile 4. alt görevler için puanlar aşağıda gösterilmiştir.

Q_s	55	56	60	70	80	100	150	10000
score_3	15	14	13	11	10	8	6	3

Q_s	2597	2800	3000	4000	5000	6000	8000	10000
score_4	65	58	53	35	26	21	14	11

Örnekler

Değerlendirici	Çözüm
1 5	
	? 0 2 4
3	
	? 3 0 1
3	
	? 0 4 3
5	
	! 3 1 0 2 4

Açıklama

Örnek girdi, $N = 5$ konuklu bir test durumu ($T = 1$) içerir. Bu test durumundaki gizli konuk konfigürasyonu Şekil 1 kısmına karşılık gelir.

Örnek çözüm tarafından yapılan ilk sorgu 0, 2, 4 şeklindedir. Bu sorguya verilen 3 cevabı, bu konukların bilinmeyen bir sırada, yan yana üç ardışık koltukta oturduklarını bize söyler.

İkinci sorguya verilen 3 cevabı, 3, 0 ve 1 numaralı konuklar hakkında da aynı şeyi söyler.

Artık 0 numaralı konuğun ortada, 2 ve 4 numaralı konukların bir tarafta, 1 ve 3 numaralı konukların ise diğer tarafta oturması gerektiğini çıkarabiliriz.

Üçüncü sorgudan sonra, konukların ya $[3, 1, 0, 2, 4]$ sırasında ya da ters sırada $[4, 2, 0, 1, 3]$ oturduklarından emin olabiliriz. İkisinden birini çıktı olarak verebiliriz.

CMS'de Kod Şablonları ve Değerlendirme Detayları

C++ ve Python için sağlanan kod şablonlarını kullanmanızı şiddetle tavsiye ederiz. Bunlar, değerlendirici ile iletişimin başarılı olup olmadığını kontrol eder ve başarılı olmadığında düzgün bir şekilde sonlanır.

Sağlanan şablonları kullanmazsanız, çözümünüzün yanlış olduğu durumlarda CMS yanlış bir sonuç (verdict) gösterebilir. Örneğin, "Output isn't correct" yerine "Execution killed by signal" veya "Execution timed out (wall clock limit exceeded)" alabilirsiniz.

Ayrıca çözümünüzü göndermeden önce yerel olarak test etmek için test aracını (aşağıya bakın) kullanmanızı öneririz. Test aracı, çözümünüzün çıktılarını kontrol eder ve geçersiz sorguların kullanımını raporlar.

Test Aracı

Çözümünüzün test edilmesini kolaylaştırmak için, CMS'den indirebileceğiniz basit bir araç sağlıyoruz. Aracı kullanmak isteğe bağlıdır. CMS'deki resmi değerlendiricinin test aracından farklı olduğunu unutmayın.

Aracı kullanmak için bir girdi dosyasına ihtiyacınız var. Sağlanan örnek girdi `seatingplan.input0.txt` dosyasını kullanabilir veya kendinizinkini oluşturabilirsiniz. Girdi dosyası, test durumu sayısını T içeren bir satırla başlamalı ve ardından her test durumu için iki satıra sahip olmalıdır: N sayısını içeren bir satır ve ardından $g_0, g_1, ..., g_{N-1}$ sayılarını içeren bir satır.

Python programları için (genellikle `pypy3 seatingplan.py` şeklinde çalıştırılır), test aracını şu şekilde çalıştırın:

```
python3 testing_tool.py pypy3 seatingplan.py < seatingplan.input0.txt
```

C++ programları için, önce çözümünüzü derleyin:

```
g++ -DEVAL -std=gnu++20 -O2 -pipe -static -s -o seatingplan seatingplan.cpp
```

ve ardından test aracını çalıştırın:

```
python3 testing_tool.py ./seatingplan < seatingplan.input0.txt
```