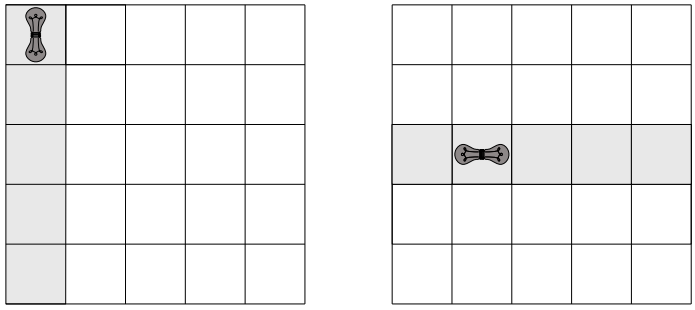
C. Ampuller

Problem Adı	Ampuller
Zaman Limiti	4 saniye
Bellek Limiti	1 gigabyte

1891 yılında Eindhoven'da ampul şirketini kurduktan kısa bir süre sonra, Frederik Philips büyük bir keşif yaptı: yatay veya dikey yönde sonsuz bir ışın yayan ampuller. Bu yeni keşfiyle modern evlerin iç tasarımında devrim yaratmak istiyordu.

Oğlu Gerard ile birlikte ayrıntılı bir kurulum planlarlar. Bir odada $N \times N$ ızgara (grid) oluşturacak şekilde N^2 lamba kurarlar. Elektrik tasarrufu yapmak için mümkün olan en az sayıda lambayı açarak tüm odayı aydınlatmak istiyorlar. Her lamba ya dikeydir, yani bulunduğu sütundaki tüm kareleri aydınlatır, ya da yataydır, yani bulunduğu satırdaki tüm kareleri aydınlatır.

Aşağıdaki şekil bir dikey lamba (solda) ve bir yatay (sağda) lamba örneği göstermektedir.



Maalesef, lambaları kurarken dikkat etmediler ve hangi lambaların yatay ya da dikey yandığını hatırlamıyorlar. Bunun yerine, odanın tamamını aydınlatmak için hangi lambaları kullanacaklarını anlamak için bazı deneyler yapıyorlar. Gerard lambaların bulunduğu odada kalırken, Frederik başka bir odadan anahtarları (açma kapama anahtarı - switch) çalıştırıyor.

Her deneyde, Frederik her bir lambayı açıyor veya kapatıyor ve Gerard kaç karenin toplamda aydınlandığını rapor ediyor; iki veya daha fazla farklı lamba tarafından aydınlatılan bir kare yalnızca bir kez sayılıyor. Deneyler sırasında kaç lambanın açıldığı önemli değil, ancak aceleleri var ve ideal olarak mümkün olduğunca az deney yapmak istiyorlar.

Onlara, odanın tamamını aydınlatan ve en az lambayı kullanan bir düzenleme bulmalarında yardımcı olun. En fazla 2000 deney yapabilirler. Ancak, daha az deney kullanırlarsa daha yüksek puan alırsınız.

Etkileşim

Bu etkileşimli bir problemdir.

- Programınız, ızgaranın yüksekliği ve genişliği olan tam sayı N'yi okuyarak başlamalıdır.
- Daha sonra, programınız değerlendirici (grader) ile etkileşime geçmelidir. Bir deney yapmak için, önce bir soru işareti "`?`" içeren bir satır yazdırmalısınız.

  ```
  Sonraki $N$ satırda, hangi lambaların yandığını belirten bir $N \times N$ ızgara yazdırın.
  ```

  Spesifik olarak, bu satırların her birinde, 0'lardan (kapalı) ve 1'lerden (açık) oluşan uzunluğu N olan bir dizi (string) yazdırın. Daha sonra, programınız belirtilen lambaları açarak aydınlatılan kare sayısını ifade eden bir tam sayı ℓ ($0 \leq \ell \leq N^2$) okumalıdır.

- Cevap vermek istediğinizde, bir ünlem işareti "`!`" içeren bir satır yazdırmalı ve ardından yukarıdaki formatta N satır ile ızgarayı yazdırmalısınız. Cevabınızın kabul edilmesi için, **lambalar tüm ızgarayı aydınlatmalı ve açılan lambaların sayısı mümkün olan en az sayıda olmalıdır**.

Bundan sonra, programınız sonlanmalıdır.

Değerlendirici uyumlu değildir (non-adaptive), yani lamba ızgarası etkileşim başlamadanbelirlenir.

Her deneyi yaptıktan sonra standart çıktıyı temizlediğinizden emin olun; aksi takdirde, programınız "Süre Aşımı" olarak değerlendirilebilir. Python'da, `input()` kullanarak satırları okuduğunuz sürece bu otomatik olarak gerçekleşir. C++'da, `cout << endl;` bir satır sonu yazdırmanın yanı sıra çıktıyı da temizler; `printf` kullanıyorsanız, `fflush(stdout)` kullanın.

Kısıtlar ve Puanlama

- $3 \leq N \leq 100$.
- En fazla 2000 deney yapabilirsiniz (son cevabı yazdırmak bir deney olarak sayılmaz). Bunu aşarsanız, "Wrong Answer" ("Yanlış Cevap") kararı alırsınız.

Çözümünüz, her biri belirli bir puan değerinde olan bir dizi test grubunda test edilecektir. Her test grubu bir dizi test durumu içerir. Bir test grubunun puanlarını alabilmek için, test grubundaki tüm test durumlarını çözmeniz gerekmektedir.

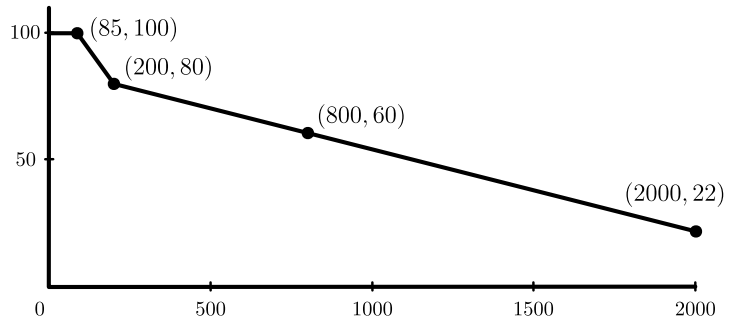
Grup	Puan	Limitler
1	11	$N = 3$
2	11	$N \leq 10$
3	78'e kadar	Ek kısıt yoktur

Son test grubunda, **puanınız yaptığınız deney sayısına bağlı olacak** ve aşağıdaki formül ile hesaplanacaktır:

$$\text{score} = \begin{cases} (2000 - Q) \cdot 29/900 & \text{if } 200 \leq Q \leq 2000, \\ 58 + (200 - Q) \cdot 4/23 & \text{if } 85 \leq Q \leq 200, \\ 78 & \text{if } Q \leq 85, \end{cases}$$

Burada Q, herhangi bir test durumunda kullanılan maksimum deney sayısıdır. Puan altındaki en yakın tam sayıya yuvarlanacaktır (rounded down).

Aşağıdaki grafik, tüm test gruplarını çözerse programınızın Q'ya bağlı olarak alacağı puanı göstermektedir. Bu problemden tam 100 puan almak için, her bir test durumunu en fazla 85 deney kullanarak çözmelisiniz.



Test Aracı

Çözümünüzü test etmeyi kolaylaştırmak için indirebileceğiniz basit bir araç sağlıyoruz. Kattis problem sayfasının altındaki “ekler” bölümüne bakın. Bu aracı kullanmak isteğe bağlıdır. Kattis'teki resmi değerlendirici (grader) programının test aracından farklı olduğunu unutmayın.

Aracı kullanmak için, “sample1.in” gibi bir giriş dosyası oluşturun. Bu dosya, bir tam sayı N ile başlamalı ve ardından ızgarayı belirten N satır içermelidir. Burada `V`, lambanın sütununu aydınlattığını ve `H`, lambanın satırını aydınlattığını belirtir. Örneğin:

```
5
VVHVH
HVHHV
VHHVV
HHHVH
HHVVV
```

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

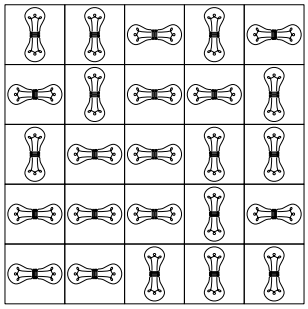
```
python3 testing_tool.py pypy3 solution.py < sample1.in
```

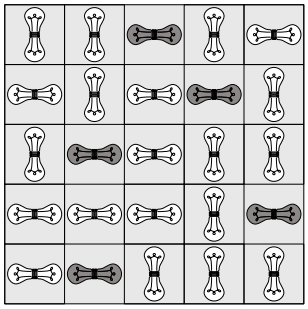
C++ programları için, önce derleyin: (örneğin `g++ -g -O2 -std=gnu++20 -static solution.cpp -o solution.out`) ve sonra çalıştırın:

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

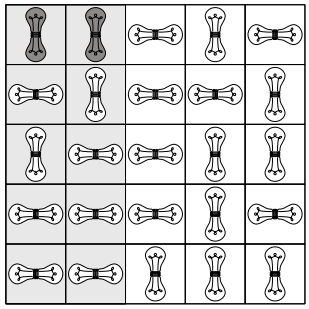
Örnek

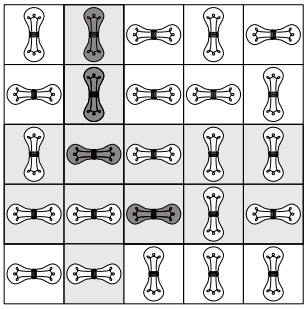
Örnek etkileşimde, program ızgara boyutunu $N = 5$ olarak okuyarak başlar. Aşağıdaki şekil, gizli ızgarayı (programın bilmediği) ve tüm ızgarayı aydınlatmak için beş lamba kullanarak verilebilecek birçok olası cevaptan birini göstermektedir. İşaretlenmiş lambalar açık durumdadır ve koyu renkli kareler aydınlatılmıştır.





Program, aşağıda gösterildiği gibi iki deney gerçekleştirir. İlk deneyde, sol üst köşedeki iki dikey lamba kullanılarak toplamda 10 kare aydınlatılır. İkinci deneyde ise toplamda 13 kare aydınlatılır. Son olarak, program cevabını (yukarıda gösterildiği gibi) yazar ve çıkar."





grader çıktısı	sizin çıktınız
5	
	? 11000 00000 00000 00000 00000
10	
	? 01000 01000 01000 00100 00000
13	
	! 00100 00010 01000 00001 01000