Kutuyu Bul

Problem İsmi	Kutuyu Bul
Zaman Sınırı	1 saniye
Hafıza Sınırı	1 gigabyte

Maj, Lund Üniversitesinde çalışan bir robotik araştırmacısıdır. Üniversitenin mahzeninde değerli bir hazine olduğunu öğrenmiştir. Hazine, yerin derinliklerindeki boş bir odada bulunan bir kutunun içindedir. Ne yazık ki Maj öylece gidip kutuyu arayamaz. Mahzen çok karanlıktır ve oraya ışıkla gitmek şüphe uyandırır. Hazineyi bulmanın tek yolu, mahzende yaşayan bir robot elektrikli süpürgeyi uzaktan kontrol etmektir.

Mahzen $H \times W$ 'lik bir ızgara şeklindedir. Bu ızgarada satırlar, 0'dan $H-1$'a (yukarıdan aşağıya) ve sütunlar 0'dan $W-1$'a (soldan sağa) şeklinde sıralanır. Yani, sol üstteki hücre $(0,0)$ ve sağ alttaki hücre $(H-1, W-1)$'dır. Hazinenin olduğu kutu $(0,0)$ haricinde bilinmeyen bir hücrededir. Robot elektrikli süpürge her gece sol üst köşeden çalışmaya başlar ve mahzen içinde hareket eder.

Maj her gece robota “<”, “>”, “^” ve “v” karakterlerinden oluşan bir string şeklinde nasıl hareket etmesi gerektiğine dair talimatlar verebilir. Daha biçimsel olarak, eğer robot her tarafı açık olan (r, c) hücresinin üzerinde duruyorsa, "<" robotu sola yani $(r, c-1)$ hücresine, “>” robotu sağa yani $(r, c+1)$ hücresine, “^” robotu yukarı yani $(r-1, c)$ hücresine ve “v”, robotu aşağı yani $(r+1, c)$ hücresine hareket ettirir.

Mahzen duvarları sağlamdır, bu nedenle robot ızgaranın dışına çıkmaya çalışırsa hiçbir şey olmaz. Kutu da sağlamdır ve itilemez. Her gecenin sonunda robot konumunu bildirir ve sol üst köşeye geri döner.

Vakit nakittir (yani zaman çok önemlidir), bu yüzden Maj kutuyu mümkün olan en kısa sürede bulmalıdır.

Etkileşim

Bu etkileşimli (interaktif) bir problemdir.

- Programınız H ve W olmak üzere iki tamsayı içeren bir satırı okuyarak başlamalıdır: Bunlar, ızgaranın yüksekliği ve genişliğidir.

- Ardından, programınız grader ile etkileşime girmelidir. Her etkileşim turunda, "`?`" soru işareti ve ardından boş olmayan bir s stringi yazdırmalısınız. Bu string şu karakterleri içerir: "`<`", "`>`", "`^`", "`v`". Bu stringin uzunluğu en fazla $20\,000$ olabilir. Daha sonra, programınız, ilgili komutları çalıştırdıktan sonra robotun konumunu gösteren iki tamsayı r, c ($0 \le r \le H-1$, $0 \le c \le W-1$) okumalıdır. Robotun her sorgudan sonra her zaman $(0,0)$'a geri döndüğünü unutmayın.
- Kutunun yerini bildiğiniz zaman, "`!`" ifadesini ve ardından kutunun satırı ve sütununu gösteren iki tam sayı r_b, c_b yazdırın ($0 \le r_b \le H-1$, $0 \le c_b \le W-1$). Bundan sonra başka sorgulama yapmadan programınız sonlanmalıdır. Bu son çıktı, puanınızı belirlerken bir sorgu olarak sayılmaz.

Bir sorgu yayınladıktan sonra standart çıktıyı temizlediğinizden (flush ettiğinizden) emin olun, aksi takdirde programınız Zaman Sınırı Aşıldı olarak değerlendirilebilir. Python'da `print()` otomatik olarak temizler. C++'da, yeni satır yazdırmaya ek olarak `cout << endl;` da temizler; printf kullanıyorsanız `fflush(stdout)` kullanın.

Grader etkileşimli değildir, yani kutunun konumu etkileşim başlamadan önce belirlenir.

Sınırlar ve Puanlama

- $1 \le H, W \le 50$.
- Kutu asla $(0,0)$ konumunda bulunmayacaktır. Bu, $H + W \ge 3$ olduğu anlamına gelir.
- Her sorgu en fazla $20\,000$ komuttan oluşabilir.
- En fazla $2\,500$ sorgu yapabilirsiniz (Son cevabı yazmanız bir sorgu olarak sayılmaz..

Çözümünüz bir dizi test senaryosu üzerinde test edilecektir. Çözümünüz bu test durumlarının *herhangi birinde* başarısız olursa (örn. yanlış kutu konumunu vermesi (Yanlış Cevap - Wrong Answer), programın çökmesi (Çalışma Zamanı Hatası - Runtime Error), zaman sınırını aşması (Zaman Sınırı Aşıldı - Time Limit Exceeded) vb.), 0 puan alırsınız ve buna uygun değerlendirme olur.

Programınız *tüm* test senaryolarında kutunun konumunu başarılı bir şekilde bulursa, AC kararını ve aşağıdaki şekilde hesaplanan puanı alırsınız:

$$\text{score} = \min\left(\frac{100\sqrt{2}}{\sqrt{Q}}, 100\right) \text{ points},$$

burada Q, herhangi bir test senaryosunda kullanılan maksimum sorgu sayısıdır. Nihai cevabın yazdırılması sorgu olarak sayılmaz. Puan en yakın tamsayıya yuvarlanacaktır.

Özellikle, 100 puan almak için programınızın her test senaryosunu en fazla $Q = 2$ sorgu kullanarak çözmesi gerekir. Aşağıdaki tablo bazı Q değerlerini ve ilgili puanı göstermektedir.

Q	**2**	**3**	**4**	**5**	**...**	**20**	**...**	**50**	**...**	**2500**
Puan	100	82	71	63	...	32	...	20	...	3

Test Aracı

Çözümünüzün test edilmesini kolaylaştırmak için size indirebileceğiniz basit bir araç sunuyoruz. Kattis problem sayfasının altındaki "eklere" bakın. Aracın kullanımı isteğe bağlıdır ve onu değiştirebilirsiniz. Kattis'teki graderın test aracından farklı olduğunu unutmayın.

Örnek kullanım ($H = 4$, $W = 5$ ve $r = 2$, $c = 3$ konumunda gizlenmiş kutu ile):

Python programı için "solution.py" deyin (normalde `pypy3 solution.py` olarak çalışır):

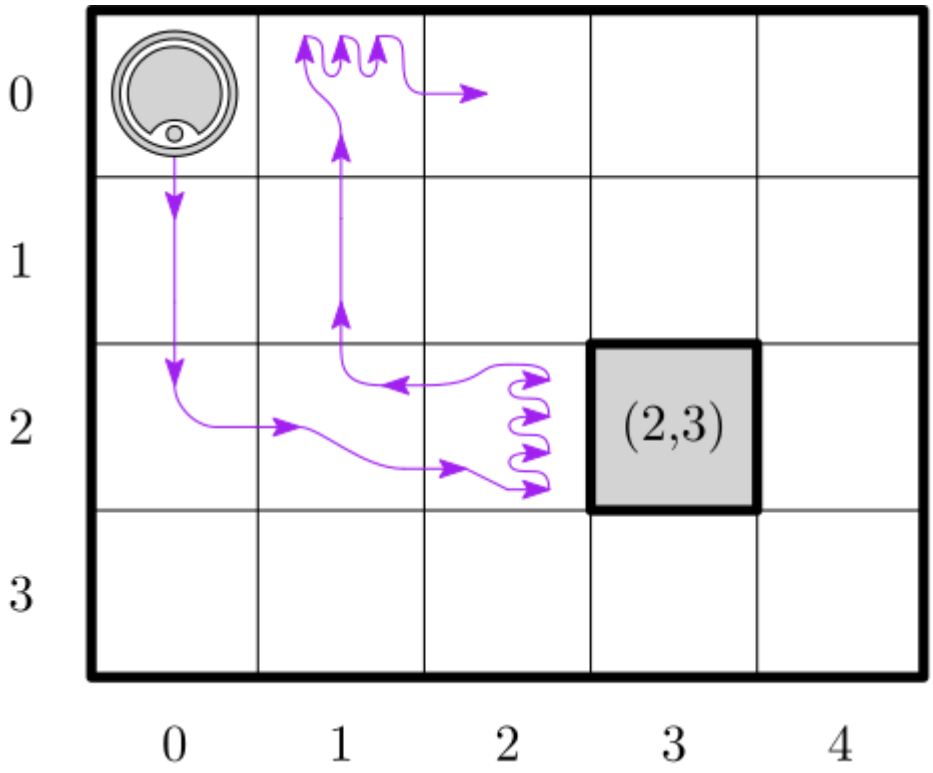
```
python3 testing_tool.py pypy3 solution.py <<<"4 5 2 3"
```

C++ programları için, ilk olarak derleyin (yani `g++ -g -O2 -std=gnu++17 -static solution.cpp -o solution.out`) ve sonra çalıştırın:

```
python3 testing_tool.py ./solution.out <<< "4 5 2 3"
```

Örnek

Şu şekildeki örnek test senaryosunu düşünün. Tablonun yüksekliği $H = 4$ ve genişliği $W = 5$'dir ve kutu $(r, c) = (2, 3)$ konumundadır. Aşağıdaki şekil robotun ilk "`? vv>>>>>><^^^^^>`" sorgusunu takip ettiğindeki yolunu gösterir. Bu da robotun $(r, c) = (0, 2)$ konumuna gelmesiyle sonuçlanır. İkinci sorgudan önce, robot tekrar sol üst köşeye $(0, 0)$ geri dönecektir. Ardından çözüm başka bir sorgu "`? >>>>>>>>vvvvvvvvvv` uygular. Bunun için robotun sağ alt köşeye gelir: $(r, c) = (3, 4)$. Şimdi çözüm cevabı "`! 2 3`" yazarak tahmin etmeye karar verir ki bu kutunun doğru konumudur.



grader çıktısı	sizin çıktınız
4 5	
	? vv>>>>>><^^^^^>
0 2	
	? >>>>>>>>vvvvvvvvv
3 4	
	! 2 3