Paint By Numbers

Paint By Numbers iyi bilinen puzzle oyunudur. Bu puzzle-ın basit bir boyutlu versiyonunu düşünelim. Bu puzzle-da, oyuncuya bir satırda dizilmiş n tane cell verilir . Cell-ler 0-dan $n-1$ -e kadar soldan sağa numaralandırılmış. Oyuncunun her cell-i siyah veya beyazla boyaması gerekiyor.Siyah cell için 'X' beyaz cell içinse '_' kullanıyoruz.

Oyuncuya k tane posıtıf tamsayıdan oluşan $c = [c_0, \ldots, c_{k-1}]$ dizisi veriliyor: *ipuçları*. Cell-leri öyle boyamalı ki siyah cell-ler bu satırda k tane ardarda gelen siyah cell-lerden oluşan parça (block) oluşturmalı. Dahası, soldan i -nji blockdaki (0 -based) siyah cell sayısı c_i -e eşit olmalı. Mesela, eğer ipuçları $c = [3, 4]$ ise, puzzle-ın çözümünde tam olarak iki tane siyah block (siyah cell-lerin ardarda sıralanmış hali) olmalı: birisi 3 diğeri 4 cell-den oluşan. Dolayısıyla, eğer $n = 10$ ve $c = [3, 4]$ ise, bir doğru çözüm "_XXX__XXXX" olur. Dikkat edin "XXXX_XXX__" doğru değil: siyah-celler blockları doğru sırada değil. Aynı şekilde, "__XXXXXXX_" doğru değil: sadece bir tane siyah-cell block var, iki ayrı blocklar değil.

Size yarım çözülmüş bir Paint By Numbers puzzle veriliyor. Yani, n ve c -yi biliyorsunuz, artı bazı cellerin siyah diğer bazılarınınsa beyaz olmaları gerektiğini de biliyorsunuz. Sizin işiniz, daha fazla bilgi elde etmelisiniz (bulmalısınız). Daha doğrusu, her doğru çözümde hangi celler siyah hangileri beyaz olacağını bulmalısınız. En az bir doüru çözğm olduğunu farzedebilirsiniz.

Uygulama detayları

Şöyle fonksiyon geliştirmelisiniz:

- `string solve_puzzle(string s, int[] c)`.
  - `s`: string of length n . For each i ($0 \le i \le n-1$) character i is:
    - 'X', eğer cell i siyah olmalıysa,
    - '_', eğer cell i beyaz olmalıysa,
    - '.', eğer i cell hakkında hiç bilgi yoksa.
  - `c`: ipuçlarını içeren k uzunlukta dizi, yukarıda tanımlandığı gibi,
  - foksiyon n uzunlukta bir dizi oluşturmalı. Her i için ($0 \le i \le n-1$) character of the output dizi-deki i karakteri şöyle olmalı:
    - 'X', eğer i cell her geçerli çözümde siyahsa,
    - '_', eğer i cell her geçerli çözümde beyazsa,
    - '?', aksi halde (i.e., iki geçerli çözüm varsa ve i cell birisinde beyaz diğerinde siyahsa).

C-de fonksiyon göstergesi biraz farklı:

- `void solve_puzzle(int n, char* s, int k, int* c, char* result)`

- `n`: length of the string `s` dizisinin uzunluğu (cell sayısı),
- `k`: `c` array uzunluğu (ipucu sayısı),
- diğer parametreler yukarıdakiyle aynı,
- n karakterli dizi oluşturmak yerine, fonksiyon çözümü `result` dizisine yazmalı.

Examples

Example 1

```
solve_puzzle(".........", [3, 4])
```

These are all possible valid solutions of the puzzle:

- "XXX_XXXX__",
- "XXX__XXXX_",
- "XXX___XXXX",
- "_XXX_XXXX_",
- "_XXX__XXXX",
- "__XXX_XXXX".

Gördüğünüz gibi 2, 6, and 7 indeksli celler (0-based) her geçerli çözümde siyah. Diğer cellerden de siyah olanlar olabilir, ama lmak zorunda değil. Dolayısıyla, doğru çözüm "??X???XX??" olur.

Example 2

```
solve_puzzle(".......", [3, 4])
```

In this example the entire solution is uniquely determined and the correct answer is "XXX_XXXX".

Example 3

```
solve_puzzle("..._._....", [3])
```

In this example we can deduce that cell 4 must be white as well — there is no way to fit three consecutive black cells between the white cells at indices 3 and 5. Hence, the correct answer is "???___????".

Example 4

```
solve_puzzle(".X.......", [3])
```

There are only two valid solutions that match the above description:

- "XXX_______",
- "_XXX______".

Thus, the correct answer is "?XX?_______".

Subtasks

In all subtasks $1 \leq k \leq n$, and $1 \leq c_i \leq n$ for each $0 \leq i \leq k-1$.

1. (7 points) $n \leq 20$, $k = 1$, s contains only '.' (empty puzzle),

2. (3 points) $n \leq 20$, s contains only '.',
3. (22 points) $n \leq 100$, s contains only '.',
4. (27 points) $n \leq 100$, s contains only '.' and '_' (information only about white cells),
5. (21 points) $n \leq 100$,
6. (10 points) $n \leq 5\,000$, $k \leq 100$,
7. (10 points) $n \leq 200\,000$, $k \leq 100$.

Sample grader

The sample grader reads the input in the following format:

- line 1: string s,
- line 2: integer k followed by k integers $c_0, \ldots, c_{k-1}$.