Pintar com Números

Pintar com Números é um quebra-cabeça bem conhecido. Vamos considerar uma versão simples, unidimensional, desse quebra-cabeças. Nesta versão, o jogador recebe uma linha com n células. As células são numeradas de 0 a $n-1$, da esquerda para a direita. O jogador tem que pintar cada célula de preto ou de branco. Vamos usar 'X' para denotar as células pretas e '_' para denotar as células brancas.

O jogador recebe uma sequência $c = [c_0, \ldots, c_{k-1}]$ de k inteiros positivos: as *dicas*. As células devem ser pintadas de tal forma que as células pretas formam exatamente k blocos de células consecutivas. Além disso, o número de células pretas do i-ésimo bloco contados da esquerda para a direita deve ser igual a c_i (os blocos são numerados a partir de 0). Por exemplo, se as dicas forem $c = [3, 4]$, a solução precisa ter exatamente dois blocos com células pretas consecutivas: um de comprimento 3 e então um outro de comprimento 4. Portanto, se $n = 10$ e $c = [3, 4]$, uma solução que satisfaz as dicas é "_XXX__XXXX". Note que "XXXX_XXX__" não satisfaz as dicas porque os blocos de células pretas não estão na ordem correta. Da mesma forma, "__XXXXXXX_" não satisfaz as dicas porque existe um único bloco de células pretas, e não dois blocos separados.

Você receberá um quebra-cabeças de Pintar por Números parcialmente resolvido. Isto é, você conhece n e c, e você também sabe que algumas células devem ser pretas e algumas devem ser brancas. Sua tarefa é deduzir mais informação a respeito das células.

Mais precisamente, uma *solução válida* deve satisfazer todas as dicas e também concordar com as cores das células conhecidas. Seu programa deve determinar quais células devem ser pretas em todas as soluções válidas, e quais células devem ser brancas em todas as soluções válidas.

Você pode supor que a entrada é tal que sempre há pelo menos uma solução válida.

Detalhes de implementação

Você deve implementar a seguinte função (método):

- `string solve_puzzle(string s, int[] c)`.
  - `s`: cadeia de comprimento n. Para cada i ($0 \leq i \leq n-1$), o caractere i é:
    - 'X', se a célula i deve ser preta,
    - '_', se a célula i deve ser branca,
    - '.', se não há informação a respeito da célula i.
  - `c`: vetor de comprimento k contendo as dicas, como definido acima,

- a função deve retornar uma cadeia de comprimento n. Para cada i ($0 \leq i \leq n-1$) o i-ésimo caractere da saída deve ser:
  - 'X', se a célula i é preta em todas as soluções válidas,
  - '_', se a célula i é branca em todas as soluções válidas,
  - '?', nos demais casos (i.e., se existem duas soluções válidas tais que a célula i é preta em uma delas e branca na outra).

Na linguagem C a assinatura da função é um pouco diferente:

- `void solve_puzzle(int n, char* s, int k, int* c, char* result)`
  - `n`: comprimento da cadeia `s` (número de células),
  - `k`: comprimento do vetor `c` (número de dicas),
  - os demais parâmetros são como os acima,
  - ao invés de retornar uma cadeia com n caracteres, a função deve escrever o resultado na cadeia `result`.

Os códigos ASCII dos caracteres usados neste problema são:

- 'X': 88,
- '_': 95,
- '.': 46,
- '?': 63.

Por favor, use os ficheiros modelo para detalhes de implementação na sua linguagem de programação.

Exemplos

Exemplo 1

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

Estas são todas as soluções corretas para o quebra-cabeças:

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

Podemos notar que as células com índices 2, 6, and 7 (a partir de 0) são pretas em todas as soluções corretas. Cada uma das outras células pode ser preta, mas não tem que ser preta. Portanto, a resposta correta é "??X???XX??".

Exemplo 2

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

Neste exemplo há apenas uma solução válida e a resposta correta é "XXX_XXXX".

Exemplo 3

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

Neste exemplo podemos deduzir que a célula 4 deve ser branca — não há como encaixar três células pretas consecutivas entre as células brancas de índices 3 e 5. Portanto, a resposta correta é "`???___????`".

Exemplo 4

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

Há apenas duas soluções válidas que satisfazem a descrição acima:

- "`XXX_______`",
- "`_XXX______`".

Portanto, a resposta correta é "`?XX?______`".

Subtarefas

Em todas as subtarefas $1 \leq k \leq n$, e $1 \leq c_i \leq n$ para cada $0 \leq i \leq k-1$.

1. (7 pontos) $n \leq 20$, $k = 1$, s contém apenas '`.`' (quebra-cabeças vazio),
2. (3 pontos) $n \leq 20$, s contém apenas '`.`',
3. (22 pontos) $n \leq 100$, s contém apenas '`.`',
4. (27 pontos) $n \leq 100$, s contém apenas '`.`' e '`_`' (apenas informação sobre células brancas),
5. (21 pontos) $n \leq 100$,
6. (10 pontos) $n \leq 5\,000$, $k \leq 100$,
7. (10 points) $n \leq 200\,000$, $k \leq 100$.

Corretor exemplo

O corretor exemplo lê a entrada no seguinte formato:

- linha 1: cadeia s,
- linha 2: inteiro k seguido de k inteiros $c_0, \ldots, c_{k-1}$.