Pintando con Números

Pintando con Números es un juego de ingenio muy conocido. Vamos a considerar una versión sencilla del mismo en una dimensión. En este juego, el jugador recibe una fila de n celdas. Las celdas están numeradas desde 0 hasta $n-1$ de izquierda a derecha. El jugador debe pintar cada celda de blanco o de negro. Utilizamos 'X' para denotar a las celdas negras y '_' para denotar a las celdas blancas.

El jugador también recibe una secuencia $c = [c_0, \ldots, c_{k-1}]$ de k enteros positivos: las *pistas*. Debe pintar las celdas de forma tal que las celdas negras de la fila formen exactamente k bloques de celdas consecutivas. Más aún, la cantidad de celdas negras en el i-ésimo bloque (Índices comenzando desde 0) desde la izquierda deberá ser igual a c_i. Por ejemplo, si las pistas son $c = [3, 4]$, el juego resuelto debe tener exactamente dos bloques de celdas negras consecutivas: uno de longitud 3 y otro de longitud 4. Por lo tanto, si $n = 10$ y $c = [3, 4]$, una solución que satisface las pistas es "_XXX__XXXX". Notar que "XXXX_XXX__" no las satisface: los bloques de celdas negras no están en el orden correcto. Además, "__XXXXXXX_" no las satisface: hay un único bloque de celdas negras, y no dos bloques separados.

Usted recibe un juego de Pintando con Números parcialmente resuelto. Es decir, conoce n y c, y adicionalmente sabe que algunas celdas en particular deben ser negras, y que algunas celdas en particular deben ser blancas. Su tarea es deducir información adicional sobre las celdas.

Específicamente, una *solución válida* es una que satisface las pistas, y también respeta los colores de las celdas ya conocidas. Su programa deberá encontrar celdas que estén pintadas de negro en toda solución válida, y celdas que estén pintadas de blanco en toda solución válida. Puede asumir que la entrada es tal que siempre existe al menos una solución valida.

Detalles de implementación

Debe implementar la siguiente función:

- `string solve_puzzle(string s, int[] c)`
  - `s`: cadena de longitud n. Para cada i ($0 \leq i \leq n-1$) el caracter i es:
    - 'X', si la celda i debe ser negra,
    - '_', si la celda i debe ser blanca,
    - '.', si no hay información sobre la celda i.
  - `c`: arreglo de longitud k con las pistas, definidas anteriormente.
  - la función debe retornar una cadena de longitud n. Para cada i ($0 \leq i \leq n-1$) el caracter i de la cadena de salida deberá ser:

    - 'X', si la celda i es negra en toda solución válida.
    - '_', si la celda i es blanca en toda solución válida.
    - '?', en otro caso (o sea, si existen dos soluciones válidas de manera tal que la celda i es negra en una de ellas y blanca en la otra).

En el lenguaje C la signatura de la función es levemente diferente:

- `void solve_puzzle(int n, char* s, int k, int* c, char* result)`
    - `n`: longitud de la cadena `s` (número de celdas),
    - `k`: longitud del arreglo `c` (número de pistas),
    - los otros parámetros son como antes.
    - en lugar de retornar una cadena de n caracteres, la función deberá escribir la respuesta en la cadena `result`.

Los códigos ASCII de los caracteres utilizados en este problema son:

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

Por favor utilice los archivos de ejemplo provistos, que contienen los detalles de implementación en su lenguaje de programación.

Ejemplos

Ejemplo 1

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

Estas son todas las posibles soluciones válidas:

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

Podemos observar que (usando índices desde 0) las celdas con índices 2, 6, y 7 son negras en todas las soluciones válidas. Cada una de las demás celdas puede ser negra, pero no tiene que serlo necesariamente. Por lo tanto la respuesta correcta es "??X???XX??".

Ejemplo 2

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

En este ejemplo existe una única solución válida, y la respuesta correcta es "XXX_XXXX".

Ejemplo 3

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

En este ejemplo podemos deducir que la celda 4 debe ser blanca también — no hay espacio suficiente para 3 celdas negras consecutivas entre las celdas blancas en las posiciones 3 y 5. Por lo tanto la respuesta correcta es "`???_???`".

Ejemplo 4

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

Hay solamente dos soluciones válidas:

- "`XXX_____`",
- "`_XXX_____`".

Por lo tanto la respuesta correcta es "`?XX?_____`".

Subtareas

En todas las subtareas $1 \leq k \leq n$, y $1 \leq c_i \leq n$ para cada $0 \leq i \leq k-1$.

1. (7 puntos) $n \leq 20$, $k = 1$, s contiene únicamente '.' (juego vacío),
2. (3 puntos) $n \leq 20$, s contiene únicamente '.',
3. (22 puntos) $n \leq 100$, s contiene únicamente '.',
4. (27 puntos) $n \leq 100$, s contiene únicamente '.' y '_' (solamente se da información sobre celdas blancas),
5. (21 puntos) $n \leq 100$,
6. (10 puntos) $n \leq 5\,000$, $k \leq 100$,
7. (10 puntos) $n \leq 200\,000$, $k \leq 100$.

Evaluador de ejemplo

El evaluador de ejemplo lee la entrada en el siguiente formato:

- línea 1: cadena s,
- línea 2: entero k seguido de k enteros $c_0, \ldots, c_{k-1}$.