Pintando con Números

Pintando con Números es un puzzle conocido. Consideraremos una versión unidimensional de este puzzle. En él, el jugador recibe una fila de n celdas. Las celdas están numeradas del 0 al $n-1$ de la izquierda a la derecha. El jugador debe pintar cada celda de color blanco o negro. Usamos 'X' para denotar celdas negras y '_' para denotar celdas blancas.

El jugador recibe una secuencia $c = [c_0, \ldots, c_{k-1}]$ de k enteros positivos: las *pistas*. Ahora, el jugador debe pintar las celdas de manera que las celdas negras en la fila formen exactamente k bloques de celdas consecutivas. Además, el número de celdas negras en el i ésimo bloque desde la izquierda (0 indexado) debe ser igual a c_i. Por ejemplo, si las pistas son $c = [3, 4]$, el puzzle resuelto debe tener exactamente dos bloques de celdas negras consecutivas: uno de longitud 3 y uno de longitud 4. Por lo tanto, si $n = 10$ y $c = [3, 4]$, una solución que satisface las pistas es "_XXX__XXXX". Nota que "XXXX_XXX__" no satisface las pistas porque los bloques de celdas negras no están en el orden correcto. Asimismo, "__XXXXXXX_" no satisface las pistas porque hay un solo bloque de celdas negras, no dos bloque separados.

Recibirás un puzzle parcialmente resuelto. Esto es, conoces n y c, y adicionalmente sabes que algunas celdas deben ser negras o blancas. Tu tarea es deducir información adicional sobre las celdas.

Decimos que una *solución válida* es aquella que satisface las pistas, y que además coincide con las celdas que ya se conoce su color. Tu programa debe encontrar las celdas que deben pintarse de negro en todas las soluciones válidas, así como las celdas que deben pintarse de blanco en todas las soluciones válidas. Puedes suponer que la entrada es tal que al menos hay una solución válida.

Detalles de la implementación

Debes implementar la siguiente función (método):

- `string solve_puzzle(string s, int[] c)`
  - `s`: una 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 acerca de la celda i.
  - `c`: un arreglo de longitud k conteniendo las pistas, en el mismo formato que arriba,

- la función debe regresar una cadena de longitud n. Para toda i ($0 \leq i \leq n-1$), el caracter i de la cadena de salida debe ser:
  - 'X', si la celda i es negra en toda solución válida,
  - '_', si la celda i es blanca en toda solución válida,
  - '?', de lo contrario (es decir, si existen dos soluciones válidas tales que la celda i es negra en una y blanca en la otra).

En el lenguaje C la declaración de la función es un poco diferente:

- `void solve_puzzle(int n, char* s, int k, int* c, char* result)`
  - `n`: la longitud de la cadena `s` (número de celdas),
  - `k`: la longitud del arreglo `c` (número de pistas),
  - los demás parámetros son iguales que arriba,
  - en vez de regresar una cadena de n caracteres, la función debe escribir la respuesta en la cadena `result`.

Los códigos ASCII de los caracteres usados en el problema son:

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

Por favor usa las plantillas proporcionadas para más detalles sobre la implementación en tu lenguaje de programación.

Ejemplos

Ejemplo 1

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

Estas son todas las soluciones válidas posibles del puzzle:

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

Se puede observar que las celdas con los índices (0 indexados) 2, 6, y 7 son negras en toda solución válida. El resto de las celdas pueden ser negras, pero no necesariamente lo son. Entonces la respuesta correcta es "??X???XX??".

Ejemplo 2

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

En este ejemplo la solución está completamente determinada y la respuesta correcta es `XXX_XXXX`".

Ejemplo 3

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

En este ejemplo podemos deducir que la celda 4 también debe ser blanca porque no hay manera de poner tres celdas negras consecutivas entre las celdas blancas en los índices 3 y 5. Entonces la respuesta correcta es "`???___????`".

Ejemplo 4

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

Sólo hay dos soluciones válidas que satisfacen los requerimientos:

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

Entonces, la respuesta correcta es "`?XX?______`".

Subtareas

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

1. (7 puntos) $n \leq 20$, $k = 1$, s sólo contiene '.' (el puzzle está vacío),
2. (3 puntos) $n \leq 20$, s sólo contiene '.',
3. (22 puntos) $n \leq 100$, s sólo contiene '.',
4. (27 puntos) $n \leq 100$, s sólo contiene '.' y '_' (sólo hay información acerca de las 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: la cadena s,
- línea 2: el entero k seguido de los k enteros $c_0, \ldots, c_{k-1}$.