Paint by Numbers

Paint by Numbers 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*. Él 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 desde la izquierda (índices comenzando desde 0) 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 válida es "_XXX__XXXX". Notar que "XXXX_XXX__" no es una solución válida: los bloques de celdas negras no están en el orden correcto. Además, "__XXXXXXX_" no es una solución válida: hay un único bloque de celdas negras, y no dos bloques separados.

Tú recibes un juego de Paint by Numbers parcialmente resuelto. Es decir, conoces n y c, y adicionalmente sabes que algunas celdas en particular deben ser negras, y que algunas celdas en particular deben ser blancas. Tu tarea es deducir información adicional sobre las celdas.

Específicamente, una *solución válida* es una que satisface las pistas, y también concuerda con los colores de las celdas conocidas. Tú programa debe encontrar las celdas que estén pintadas de negro en toda solución válida, y las celdas que estén pintadas de blanco en toda solución válida.

Puedes asumir que la entrada es tal que siempre existe al menos una solución valida.

Detalles de implementación

Debes implementar la siguiente función:

- `string solve_puzzle(string s, int[] c)`.
  - `s`: string 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, como se definió anteriormente.
  - la función debe retornar un string de longitud n. Para cada i ($0 \leq i \leq n-1$) el caracter i del string 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 (es decir, 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 firma de la función es levemente diferente:

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

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

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

Usar los archivos con las plantillas para obtener los detalles de implementación en tu 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) $n \leq 200\,000$, $k \leq 100$.

Grader de ejemplo

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

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