Secuencias divertidas

El problema

En un juego de aventuras se tienen tres tipos de personajes: PRINCESA, CABALLERO y DRAGÓN. Se manejan, así mismo, secuencias de personajes. En dichas secuencias:

- El *elemento central* es el situado en la posición (i+j):2 (siendo : la división entera).
- La *mitad inferior* es la subsecuencia de elementos que preceden al elemento central.
- La *mitad superior* es la subsecuencia de elementos que siguen al central.

Una secuencia de personajes se considera *divertida* cuando, bien contiene menos de dos personajes, bien contiene al menos dos personajes, y se cumplen las siguientes condiciones:

- Si el número de dragones en la mitad inferior de la secuencia supera al número de caballeros en la mitad superior de la secuencia, entonces el elemento central debe ser CABALLERO.
- Si el número de dragones en la mitad inferior de la secuencia es igual al número de caballeros en la mitad superior de la secuencia, entonces el elemento central debe ser PRINCESA.
- Si el número de dragones en la mitad inferior es menor que el número de caballeros en la mitad superior de la secuencia, entonces:
  - El elemento central deberá ser DRAGON, siempre y cuando en el resto de la secuencia haya, al menos, una princesa.
  - Si en el resto de la secuencia no hay ninguna princesa, el elemento central debe ser PRINCESA.
- Por último, tanto la mitad inferior como la mitad superior son, a su vez, secuencias *divertidas*.

Debe desarrollarse un algoritmo recursivo **con coste lineal** en el peor caso que, dada una secuencia de personajes almacenada en un vector (el primer elemento de la secuencia se almacena en la posición 0, el segundo en la posición 1, etc.), determine si la secuencia es o no *divertida*. Aparte de desarrollar sistemáticamente el algoritmo (siguiendo el método explicado en clase) debe demostrarse que su complejidad es lineal, planteando y resolviendo las recurrencias adecuadas.

Trabajo a realizar

Para realizar el control se proporciona un archivo `divertidas.cpp` que contiene un programa que lee por la entrada estándar casos de prueba y los ejecuta (llamando a la función `es_divertida`). Cada caso de prueba consiste en dos líneas: La primera línea indica el tamaño *n* del vector ($0 \leq n \leq 1000000$). La segunda línea contiene los *n* valores del vector, codificados como números enteros (PRINCESA=0, CABALLERO=1, DRAGON=2) separados por espacios. La lista de casos de prueba termina con un -1. Por cada caso de prueba se escribirá una línea con `SI` si la secuencia contenida en el vector es *divertida* (y, por tanto, `es_divertida` devuelve `true`), y `NO` en caso contrario (y, por tanto, `es_divertida` devuelve `false`)

A continuación, se muestra un ejemplo de entrada / salida.

Entrada	Salida
6 0 2 1 0 0 2 3 0 2 1 9 1 2 0 1 1 0 0 0 0 8 1 2 0 1 1 0 2 0 4 2 1 0 2 7 1 0 1 0 1 0 1 1 2 0 -1	SI SI SI NO SI NO SI SI

Importante:

Deben indicarse los nombres y apellidos de los autores del control en el comentario que aparece al principio del archivo `divertidas.cpp` (**de lo contrario, el control no puntuará**). En caso de que alguno de los miembros del grupo no haya realizado el control, se deberá indicar en dicho comentario.

Únicamente puntuarán las soluciones que superen satisfactoriamente los casos de prueba del juez, en el tiempo máximo asignado para ello (veredicto de `CORRECT`).

Importante: aparte de la implementación, el algoritmo debe especificarse, diseñarse y analizarse correctamente, utilizando el método explicado en clase, utilizando los comentarios habilitados al respecto. Llevar a cabo una especificación, un diseño (generalización, casos, inmersión, terminación…) y un análisis de la complejidad correctos (recurrencias …) será esencial para superar esta prueba.

Únicamente se tendrá en cuenta el último envío `CORRECT` enviado dentro de plazo.

No modificar el código proporcionado. Únicamente deben responderse a los distintos apartados, en el interior de los comentarios, implementar la función `es_divertida` y todas las funciones auxiliares que se consideren necesarias.