Mensajes codificados

El objetivo de este control es, por una parte, trabajar con las representaciones internas de los TADs lineales añadiendo una nueva operación a un TAD existente, y, por otra, resolver un problema utilizando, entre otras, la nueva operación añadida, así como otros TADs lineales.

1) El problema

El agente 0069 ha inventado un nuevo método de codificación de mensajes secretos. El mensaje original X se codifica en dos etapas:

- X se transforma en X' reemplazando cada sucesión de caracteres consecutivos que no sean vocales por la sucesión inversa.
- X' se transforma en el mensaje X" obtenido cogiendo sucesivamente el primer carácter de X', luego el último, luego el segundo, luego el penúltimo, etc.

Por ejemplo, si el mensaje original X = *Bond, James Bond*, los resultados de las dos etapas de codificación son: X' = *BoJ ,dnameB sodn* y X" = *BnodJo s, dBneam*.

Se debe construir un programa que lea mensajes de la entrada estándar (un mensaje en cada línea), y escriba los mensajes codificados en la salida estándar (uno por línea).

Ejemplo de entrada / salida:

Entrada	Salida

`Bond, James Bond`
`La vida es bella`
`Me gusta comer naranjas`
	`BnodJo s, dBneam`
	`Laalvl eisd ab e`
	`Mseagn juatrsaarc  noem`

2) Trabajo a realizar

Se proporciona el archivo `main.cpp` en el que se implementa toda la lógica de entrada / salida necesaria, así como otras funcionalidades necesarias para resolver el problema. El código proporcionado no debe modificarse.

Hay que añadir a dicho archivo la implementación de la siguiente funcion:

```
// Reemplaza cada secuencia de caracteres no vocales consecutivos
// por su inversa. 'mensaje' se deberá modificar con el resultado
// de realizar dicho proceso de inversion.
void invierteSecuenciasNoVocales(Lista<char>& mensaje);
```

La implementación del TAD `Lista` que debe utilizarse será la basada en una lista de nodos doblemente enlazada que se proporciona junto a este enunciado en el campus virtual (dicha implementación es la vista en clase a la que se le ha incorporado una operación `print` usada en el main para visualizar la lista por pantalla). Deberá extenderse dicho TAD con una nueva operación:

- `enredar:` Debe modificar la lista intercalando sus nodos de la siguiente forma: supongamos que los nodos de la lista enlazada de izquierda a derecha son n_1; n_2; n_3; n_4; ... ; n_{k-3}; n_{k-2}; n_{k-1}; n_k , al finalizar la ejecución del método los nodos estarán colocados como n_1; n_k; n_2; n_{k-1}; n_3; n_{k-2}; n_4; n_{k-3}; n_5; .... Dicha implementación debe ser lo más eficiente posible, evitando liberar y reservar memoria, y hacer copias de los campos. Obsérvese que dicha operación puede usarse para realizar la segunda fase de codificación (el código proporcionado en `main.cpp` utiliza dicha operación).