Mensajes codificados

El objetivo de este problema 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 006 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, para cada mensaje X leído, escriba X"#X_I", donde X" es el resultado de codificar X, y X_I" es X" escrito *al revés*.

Ejemplo de entrada / salida:

Entrada	Salida
`Bond, James Bond`	`BnodJo s, dBneam#maenBd ,s oJdonB`
`La vida es bella`	`Laalvl eisd ab e#e ba dsie lvlaaL`
`Me gusta comer naranjas`	`Mseagn juatrsaarc  noem#meon craasrtauj ngaesM`

2) Trabajo a realizar

Se proporciona un archivo `main.cpp` en el que se implementa toda la lógica de entrada / salida, así como alguna lógica adicional. Hay que añadir a dicho archivo la implementación de la siguiente función:

```
// 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);
```

Se proporciona también una implementación del TAD `Lista` a la que debe añadirse una nueva operación, `enredar`, que 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 contenidos de los nodos. Obsérvese que dicha operación puede usarse para realizar la segunda fase de codificación (el código proporcionado en `main.cpp` utiliza dicha operación).

Dicha implementación incluye, además, un par de métodos adicionales para escribir la lista, una de ellas desde el principio hasta el final, y otra desde el final hasta el principio (dichas operaciones se usan en el programa principal para realizar la impresión de los mensajes codificados). Por tanto, debe utilizase dicha implementación, en lugar de la proporcionado en el campus virtual.

Con la excepción de añadir las funcionalidades pedidas, el resto del código proporcionado no debe modificarse (cualquier modificación supondrá automáticamente no puntuar en este control).

La implementación de `invierteSecuenciasNoVocales` puntuará un máximo de 5 puntos, lo mismo que la implementación de la operación `enredar`. En ambos casos se valorará la corrección, eficiencia y claridad de las soluciones (se recomienda el uso de comentarios para aumentar dicha claridad), así como el uso de buenas prácticas de programación. En ninguno de los casos se admitirán soluciones con coste superior al lineal. Asimismo, en la implementación de `enredar` no se admitirán soluciones que impliquen (directa, o indirectamente) operaciones de reserva y/o liberación de memoria, o asignaciones a los contenidos de los nodos.