Estructura de Datos y Algoritmos

Grado de Desarrollo de Videojuegos. Curso 2021-2022
Examen final. Convocatoria ordinaria Tiempo: 3 horas

Instrucciones

- La entrega se realiza en el juez automático de los laboratorios accesible desde la url `http://exacrc` (cada ejercicio en su correspondiente problema del juez, acabados respectivamente en Ej1, Ej2 y Ej3). Para acceder debes usar el usuario/contraseña que has recibido al comienzo del examen.
- Al principio de cada fichero .cpp debe aparecer, en un comentario, vuestro nombre y apellidos, dni y puesto de laboratorio. También debéis incluir unas líneas explicando qué habéis conseguido hacer y qué no.
- Todo lo que no sea código C++ (explicaciones, respuestas a preguntas, etc.) debe ir en los propios ficheros en comentarios debidamente indicados.
- Los TADs, las plantillas y ficheros de entradas de ejemplo para cada ejercicio se descargan desde `http://exacrc/EDA-Enero22.zip`.
- Podéis realizar varias entregas para un mismo ejercicio pero solamente se tendrá en cuenta la última.
- Podéis acceder a la referencia de C++ en `http://exacrc/cppreference`

Ejercicio 1 [3 puntos]

Implementa una función (**externa** a la clase) que adelante un número de posiciones dado a un segmento (elementos en posiciones consecutivas) de la lista pasada como argumento (del tipo `list` visto en clase, fichero `list_eda.h`). Por ejemplo, si la lista está formada por los elementos *a b c d e f g*, y adelantamos $k = 3$ posiciones el segmento que comienza en la posición $pos = 5$ (las posiciones se numeran desde 0 hasta n-1, siendo n el número de elementos de la lista) y tiene longitud lon=2, entonces la lista resultante es *a b* ***f g*** *c d e* (donde se han marcado en negrita los elementos desplazados). Si la posición de origen o destino del segmento no es válida (lo que incluye el caso de la lista vacía), o si $lon = 0$ o $k = 0$, la operación no tendrá efecto (ver cuarto caso). Si el segmento se sale de la lista, es decir, si $pos + lon > n$ se tomará el segmento de los últimos $n - pos$ elementos (ver segundo caso). Se puede asumir que los 4 parámetros n, pos, lon y k son no-negativos.

Se valorará la eficiencia y complejidad tanto en tiempo como en espacio del algoritmo implementado, las cuales debes indicar y justificar. En particular, se penalizarán bastante el uso de espacio no constante y los recorridos innecesarios.
La función principal proporcionada para hacer pruebas comienza leyendo el número de casos. Cada uno consta de dos líneas. En la primera aparecen los números n, pos, lon y k, y en la segunda los n elementos de la lista. Una vez leidos e insertados en la lista se llama a la función pedida con los argumentos correspondientes y se muestra por pantalla la lista modificada (ver ejemplos).

Entrada	**Salida**
4	
7 5 2 3 a b c d e f g	a b f g c d e
7 5 4 1 x y z p r s t	x y z p s t r
7 2 4 2 x y z p r s t	z p r s x y t
7 2 2 3 a b c d e f g	a b c d e f g

Cuestión: ¿Se tendría alguna ventaja al implementar la operación como método interno a la clase `list`? Justifica la respuesta.