Ejercicios de listas

OBJETIVOS

El objetivo de estos ejercicios es repasar los conceptos sobre el TAD lista en lazada.

En la página de UAMX de la asignatura dispones del esqueleto del código para poder trabajar con estos ejercicios. El examen serán dos ejercicios similares a estos. Los ejericios tienen una dificultad indicada con asteriscos.

Ejercicio ordenado (*)

Crear una función esOrdenado que devuelva 1 si la lista pasado como parámetro está ordenada de menor a mayor y 0 en caso contrario.

```
int esOrdenado(Lista L);
```

Ejemplo de ejecución: Si llamas a esOrdenado con C -> 3 -> 3 -> 5 devolverá 1 y si la llamas con C -> 3 -> 3 -> 1 devolverá 0.

Ejercicio ordenar (**)

Crear una función insertarOrdenado que añade los elementos a la lista de forma que la lista quede siempre ordenada de menor a mayor. Debe llamar a la función **`insertar`** del TAD listas

```
void insertarOrdenado(int X, Lista L);
```

Ejemplo de ejecución: Tras insertar los elementos 11, 5 y 9 en cualquier order. La lista debe quedar siempre así:

```
C-> 5-> 9-> 11
```

Ejercicio mezclar listas (***)

Implementar una función Mezclar que tenga como parámetros dos listas de enteros ordenados de menor a mayor y que devuelva una nueva lista como unión de ambas con sus elementos ordenados de la misma forma

```
Lista Mezclar(Lista L1,Lista L2);
```

Ejemplo de ejecución: Tras llamar a mezclar con las listas ordenadas C -> 1 -> 3 -> 5 y C -> 2 -> 4 -> 6. La lista resultado debe quedar así:

```
C-> 1-> 2-> 3-> 4-> 5-> 6
```

Ejercicio invertir (*)

Realizar una función que reciba como argumento una lista y cree otra lista con los datos en orden inverso.

```
Lista invertir (Lista L);
```

Ejemplo de ejecución: Tras llamar C -> 4 -> 3 -> 5 La lista resultado debe quedar así:

```
C-> 5-> 3-> 4
```

Ejercicio contar (*)

Realizar una función denominada longitud que reciba como argumento una lista y devuelva el número de elementos que contiene.

```
int longitudL(Lista L);
```

Ejemplo de ejecución: Tras llamar C -> 4 -> 3 -> 5 La función debe devolver 3

Ejercicio minrepetir (**)

Implementar una función denominada minrepetir que recibe como argumento una lista de números. Esta función debe encontrar el dato mínimo y contar el número de veces que se repite este número. El número de repeticiones de debe guardar en la variable pasada como argumento

```
int minrepetir (Lista L, int *n_repeticiones);
```

Ejemplo de ejecución: Tras llamar C -> 4 -> 3-> 3 -> 5, la función debe devolcer 3 y almacenar en la variable n_repeticiones el valor 2, que es el número de veces que aparece el 3.

Ejercicio guardar en fichero (**)

Implementa una función que guarde en fichero todos los números de la lista. Se deberán guardar los número separados por un character dado como parámetro y que guarde en cada fila del fichero como máximo un número dado por parámetro. La entrada es la lista a guardar, el nombre del fichero, el máximo de números por fila y el separador. El valor de retorno debe ser 1 si se ha podido realizar la operación y 0 si no se han podido guardar los datos.

```
int guarda_lista(Lista L, const char *nfile, int max, char sep);
```

Ejemplo. Si tenemos las lista C -> 3 -> 4 -> 10 -> 11 -> 18 y llamamos a la función del siguiente modo

```
guarda_lista(L, “fichero.txt”, 3, “|”);
```

El fichero de salida será

```
3|4|10
11|18
```

Ejercicio intercambia valores (**)

Implementa una función intercambio que dada una lista y dos valores del dato intercambie los datos en la lista sin intercambiar los nodos. Se debe llamar a la función buscarAnterior hecha en clase de teoría y a esVacio para comprobar si L está vació. La función debe devolver 1 si ha podido realizar el intercambio y 0 en caso contrario

```
int intercambia_valores(Lista L, int dato1, int dato2);
```

Ejemplo. Si tenemos las lista C -> 3 -> 4 -> 10 -> 11 -> 18 y llamamos a la función del siguiente modo

```
intercambia_nodos(L, 4, 11)
```

La lista se verá modificada como sigue

```
Cabecera -> 3 -> 11 -> 10 -> 4 -> 18
```

Ejercicio intercambia nodos (***)

Implementa una función intercambio que dada una lista y dos valores del dato intercambie los nodos de dichos datos en la lista. No debe cambiiar los valores sino los nodos. Se debe llamar a la función buscarAnterior hecha en clase de teoría y a esVacio para comprobar si L está vació. La función debe devolver 1 si ha podido realizar el intercambio y 0 en caso contrario

```
int intercambia_nodos(Lista L, int dato1, int dato2);
```

Ejemplo. Si tenemos las lista C -> 3 -> 4 -> 10 -> 11 -> 18 y llamamos a la función del siguiente modo

```
intercambia_nodos(L, 4, 11)
```

La lista se verá modificada como sigue

```
Cabecera -> 3 -> 11 -> 10 -> 4 -> 18
```