Unidad 2 - Listas

Ejercicio N°1

Implementar una función llamada "printLista" que reciba una lista y una posición **"p"** e imprima los elementos de esa lista a partir de la posición **"p"**. No debe modificarse la lista original. Ej: listaOriginal: 1->2->3->4->5->6->7->8->9->10 con p=5 Por pantalla se verá: 5->6->7->8->9->10

Ejercicio N°2

Implementar una función que recibe una lista de números enteros y un número entero **"n" y que** modifique la lista borrando todos los elementos de la lista que tengan el número "**n**". Imprimir la lista antes de hacer el llamado a la función y luego del llamado a la función para mostrar que la lista original cambió. ej: lista: 1->2->3->4->5->3->7->8->3->10 con n=3 debe pasar con la función a 1->2->4->5->7->8.

Ejercicio N°3

Implementar una función que intercambie los elementos entre 2 listas de números enteros que recibe como parámetros. Si los tamaños de las listas son distintos, igual debe cambiar los datos y cambiarían los tamaños de cada lista. Ejemplo sea **lista1**= 15->1->8->35->40->25->12 y **lista2**= 3->4->912->45->66 al ejecutar la función quedarían **lista1**= 3->4->912->45->66 y **lista2**=15->1->8->35->40->25->12.

Ejercicio N°4

Escribir un programa que pida al usuario una palabra o frase y la almacene en una Lista separando letra por letra, luego pedirá al usuario una vocal que desee contar y, por último, se debe imprimir por pantalla la lista y el número de veces que aparece la vocal en la palabra o frase. Validar que la Lista no esté vacía y que la letra a contar que introduzca el usuario sea una vocal.

Ejercicio N°5

Crear un programa que pida al usuario dar elementos a una Lista de números enteros. Luego cree una función que reciba una lista int L1 y devuelva otra lista int L2 conteniendo los elementos repetidos de L1. Por ejemplo, si L1 almacena los valores 5->2->7-> 2->5->5->1, debe construirse una lista L2 con los valores 5->2. Si en L1 no hay elementos repetidos se debe indicar al usuario que no hay elementos repetidos en L1, de lo contrario imprimir ambas listas.

Ejercicio N°6

Utilizando la estructura lista, crear un programa que gestione una lista circular de contactos. Cada contacto contiene un nombre (cadena de caracteres) y un número de teléfono (cadena de caracteres).

Tareas

1. **Agregar Contactos:**
    - Usa los métodos de `CircList` para agregar varios contactos a la lista. Deberás insertar al menos cinco contactos con nombres y números de teléfono distintos.
2. **Mostrar Contactos:**
    - Implementa una función que recorra la lista circular y muestre todos los contactos en orden. Usa el método `imprimir()` de `CircList`.
3. **Buscar un Contacto por Nombre:**
    - Implementa una función que busque un contacto por su nombre. Usa el método `getDato(int pos)` para acceder a los datos de los nodos y busca el contacto en la lista.
4. **Eliminar un Contacto por Nombre:**
    - Implementa una función que elimine un contacto por su nombre. Deberás usar los métodos `getDato(int pos)` y `eliminarPorValor(T valor)` (supón que `eliminarPorValor` está implementado, aunque no esté explícito en el código proporcionado).
5. **Contar Contactos:**
    - Usa el método `getTamanio()` para contar el número de contactos en la lista y muestra el resultado.

Ejercicio N°7

Utiliza la clase `ListaDoble` para gestionar una lista de estudiantes, donde cada estudiante tiene un nombre (cadena de caracteres) y una edad (entero). Implementa las siguientes funcionalidades:

1. **Agregar Estudiantes**:
    - Usa los métodos de `ListaDoble` para agregar varios estudiantes a la lista. Debes insertar al menos cinco estudiantes con nombres y edades diferentes.
2. **Mostrar Estudiantes**:
    - Implementa una función que recorra la lista y muestre todos los estudiantes en orden. Utiliza el método `imprimir()` de `ListaDoble`.
3. **Buscar un Estudiante por Nombre**:
    - Implementa una función que busque un estudiante por su nombre. Utiliza el método `getDato(int pos)` para acceder a los datos de los nodos y busca al estudiante en la lista.
4. **Eliminar un Estudiante por Nombre**:
    - Implementa una función que elimine un estudiante por su nombre. Utiliza los métodos `getDato(int pos)` y `remover(int pos)` para encontrar y eliminar al estudiante de la lista.
5. **Contar Estudiantes**:
    - Usa el método `getTamanio()` para contar el número de estudiantes en la lista y muestra el resultado.