INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA – CAMPUS CAMPINA GRANDE			
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA PARAÍBA Campus Campina Grande	CURSO:	CURSO DE ENGENHARIA DA COMPUTAÇÃO	
	PERÍODO:		TURMA:
	DISCIPLINA:	PROGRAMAÇÃO E ESTRUTURAS DE DADOS	
	PROFESSOR:	CÉSAR ROCHA VASCONCELOS	SEMESTRE LETIVO

Prática – LISTAS DUPLAMENTE ENCADEADAS

1) Utilizando técnicas de modularização de sistemas vistas em sala, implemente o tipo abstrato de dados **TListaDupEnc** (de implementação duplamente encadeada e tipo base inteiros). Implemente este tipo numa biblioteca chamada **listadupenc**. Esta biblioteca deve seguir o mesmo padrão da biblioteca *listaenc* desenvolvida no exercício anterior de lista encadeada. Ou seja, contendo uma interface bem definida (definição de tipos e funções) em um arquivo com extensão .h . A biblioteca deve possuir as seguintes funções básicas vistas em sala:
    a. Criação da lista duplamente encadeada vazia
    b. Verificar se a lista duplamente encadeada está vazia
    c. Obter o tamanho da lista duplamente encadeada
    d. Obter o valor do elemento de uma posição dada
    e. Obter a posição de elemento cujo valor é dado
    f. Inserir um elemento na lista duplamente encadeada, dada a sua posição
    g. Remover um elemento de uma determinada posição
    h. **Lembre-se:** em cada uma das operações, identifique possíveis situações de erros do usuário e exiba mensagens para ele nestas situações. (Ex. o programa deve exibir mensagens no caso do usuário tentar remover um item numa lista que está vazia etc.). Isto é um sinal de maturidade na programação e torna seu TAD mais robusto (menos suscetível a travamentos).

2) Modifique o menu de operações do editor da lista encadeada da prática anterior. Ou seja, reutilize o seu código para geração de um novo menu de operações contendo todas as novas operações deste TAD na questão acima. Execute e teste este novo TAD (ele deve funcionar normalmente)

3) Uma vez que as operações básicas foram implementadas na questão 1, insira agora na biblioteca **listadupenc** as seguintes **novas operações** (crie novos protótipos! Não use funções existentes):
    a. Uma função que possa retornar o **número de ocorrências** de um dado elemento numa lista
    b. Inserir um dado elemento na **primeira posição** de uma lista encadeada;
    c. Inserir um dado elemento **na última posição** de uma lista encadeada;
    d. **Modificar um elemento** de uma lista encadeada, sendo dada a sua posição e o novo valor;
    e. **Remover o primeiro** elemento de uma lista encadeada;
    f. **Remover o último** elemento de uma lista encadeada;
    g. **Remover** um elemento dado o seu valor

4) Implemente duas funções na sua biblioteca: uma primeira função que receba duas listas duplamente encadeadas e possa concatenar todos os elementos de lista2 a partir do final de lista1. A lógica de funcionamento é mostrada na figura à esquerda abaixo. O protótipo da função: `no* concatena(TListaDupEnc *lst1, TListaDupEnc *lst2 )`. A segunda função deve construir uma nova lista a partir da intercalação dos nós de outras duas listas. Esta função recebe duas listas como parâmetro e retorna uma terceira lista intercalada resultante. A lógica de funcionamento desta segunda função é mostrada na figura à direita abaixo. O protótipo: `no* merge(TListaDupEnc *lst1, TListaDupEnc *lst2 )`