CURSO: Sistemas de Informação
DISCIPLINA: Estrutura, Pesquisa e Ordenação de Dados
PROFESSOR (**A**): Frederico Brito Fernandes
SEMESTRE/ANO: 2010.2
TURMA: Noite
CARGA HORÁRIA: 80 h/a

Exercícios de Fixação 1: Listas Encadeadas

EF 1.1) Declare a estrutura eNo (struct eNo) definindo os campos (info, prox) com os tipos (int, *eNo). De preferência, ao declarar sua estrutura, crie o tipo tNo, que representa um nó da sua lista ligada (use o typedef). Crie uma lista ligada, com o nome cabeca (tNo *cabeca), contendo um nó com '*info*'=0 (0,NULL);

EF 1.2) Crie uma função para acrescentar um nó no final de uma lista ligada;
Protótipo: void insereNoFim(int valor, tNo *cabeca)

EF 1.3) Use sua função insereNoFim() do quesito anterior para acrescentar os nós com os valores (1,3,7,9), ligados nessa ordem;

EF 1.4) Crie uma função que retorne um nó da lista, buscando por um valor específico;
Protótipo: tNo buscaNo(int valor, tNo *cabeca)*

EF 1.5) Crie uma função que transforme uma lista simplesmente encadeada em uma lista simplesmente encadeada circular;
Protótipo: void transformaEmCircular(tNo cabeca)*

EF 1.6) Crie uma função que recebe uma lista ligada com a '*info*' ordenada e um valor, e que insira um nó contendo esse valor de forma que a lista continue ordenada;
*Protótipo: void insereDeFormaOrdenada(int valor, tNo *cabeca)*

EF 1.7) Crie uma função para inverter a posição do 1º com o 2º nó de uma lista;
*Protótipo: void invertePrimeiroSegundo(tNo *cabeca)*

EF 1.8) Crie uma função para criar duas listas ligadas, onde a primeira metade pertencerá a listaParteInicial, e a segunda metade em listaParteFinal. No caso de quantidade ímpar de nós, a listaParteInicial deverá ser a maior;
*Protótipo: void criaDuasMetades(tNo * cabeca, tNo * listaParteInicial, tNo * listaParteFinal)*

EF 1.9) Crie uma função para criar duas listas: listaPar, contendo nós pares da lista, e listaImpar contendo os nós ímpares;
*Protótipo: void divideEmParImpar(tNo * cabeca, tNo * listaPar, tNo * listaImpar)*

EF 1.10) Crie uma função que retorne uma lista nova, cujos nós sejam os nós, de forma invertida de outra lista
Protótipo: tNo gerarListaInvertida(tNo *cabeca)*