Estruturas de Dados e Seus Algoritmos
Lista de Exercícios - Listas Simplesmente Encadeadas

Dada a seguinte *struct* de uma lista simplesmente encadeada:

```
typedef struct lista {
     int info;
     struct lista *prox;
} TLista;
```

Responda às seguintes questões

1. Inverta uma lista simplesmente encadeada. Uma entrada do tipo 5 -> 6 -> 10 -> 1 -> 29 seria convertida em 29 -> 1 -> 10 -> 6 -> 5. Faça uma função de protótipo `TLista *inverter(TLista *L)` onde L é o ponteiro do primeiro elemento da lista.
2. Escreva um algoritmo que leia uma informação x e remova TODAS as ocorrências de nós contendo a informação x de uma lista simplesmente encadeada L. (Obviamente, a lista L pode conter nós com informação repetida.). O protótipo da função é `TLista *remove(TLista *L, int x)`
3. Em uma lista simplesmente encadeada. Para cada ocorrência do número x, trocar o seu anterior pelo seu próximo. Por exemplo, na lista original a -> x -> c -> x -> e -> f, primeiro se troca a primeira ocorrência de x, isto é c -> x -> a -> x -> e -> f, e em seguida se troca a segunda ocorrência de x, isto é c -> x -> e -> x -> a -> f. O protótipo da função deverá ser `TLista *inverte_vizinhos(TLista *L, int x)`
4. Ordenar uma lista simplesmente encadeada. O protótipo da função deve ser `TLista *ordenar(TLista *L);`
5. Considere duas listas ordenadas L1 e L2. Escreva uma função de protótipo `TLista *merge(TLista *L1, TLista *L2)` que irá retornar uma terceira lista, também ordenada, diga-se L3, que possui todos os números das listas L1 e L2.
6. Escreva uma função `TLista *inverter_mn(TLista *L, int n, int m)`. A função deve verificar se existe uma sequência de valores na lista L que começa em n e termina em m. Caso a sequência exista, inverta a ordem dos elementos que estão entre n e m nessa lista. Assuma que a lista não pode ter números repetidos. Exemplo: para n = 10 e m = 40 para a lista de entrada -20 -> 10 -> 15 -> 90 -> 40 -> 56, temos como saída -20 -> 10 -> 90 -> 15 -> 40 -> 56.