Lista de Exercícios de Estruturas de Dados e Algoritmos

Q1) Considerando as seguintes declarações de lista encadeada:

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

Escreva uma função em C que, dada uma lista **l** qualquer, inverta os elementos de **l**. O protótipo da função de inversão é o seguinte: **void inverte (TLSE* l);**

Q2) Refaça a Q1, escrevendo uma função em C que, dada uma lista **l** qualquer, inverta os elementos de **l** em uma outra lista de saída. Portanto, a lista de entrada não pode ser alterada. O protótipo da função de inversão é o seguinte: **TLSE* inverte (TLSE* l).**

Q3) Considerando a declaração da Q1, escreva uma função em C que, dada uma lista l qualquer, desloque uma vez os elementos de l, de acordo com n. Se n é ímpar, o elemento que está na última posição passa a ser o primeiro quando a lista é deslocada. Senão, o elemento que está na primeira posição passa a ser o último. O protótipo desta função é o seguinte: **TLSE* desloca (TLSE* l, int n)**.

Q4) Considere a existência de um tipo que representa um aluno numa universidade hipotética:

```
typedef struct aluno {
        int mat;
        float cr;
        struct aluno *prox;
}TAluno;
```

Escreva uma função que copie uma lista. A lista original deve permanecer inalterada. O protótipo da função é o seguinte: **TAluno *copia (TAluno *l).**

Q5) Considerando a definição de lista de Q1, escreva uma função em C que remova todas as ocorrências de um elemento numa lista. Seu protótipo O protótipo desta função é o seguinte: **TLSE* rto (TLSE* l, int elem)**.

Q6) Considerando a definição de lista de Q1, escreva uma função em C que, dada uma lista **l** qualquer, retorne, numa lista de saída, os elementos ímpares e os elementos pares da lista **l**, na ordem em que os elementos aparecem em **l**. Ao final da execução desta função, a lista resultante terá todos os elementos da lista **l** (primeiro os ímpares, depois os pares, na ordem em que eles aparecem em **l**), e a lista **l** continuará contendo os seus elementos. O protótipo da função é o seguinte: **TLSE* i_p (TLSE *l).**

Q7) Refaça Q6, alterando a lista passada como parâmetro. O protótipo desta função é o seguinte: **void* i_p (TLSE *l).**

Q8) Considerando as seguintes declarações de uma lista encadeada:

```
typedef struct lista{
        int mat;
        char nome[81];
        float cr;
        struct lista *prox;
}TLista;
```

Escreva uma função em C que, dadas duas listas l1 e l2 encadeadas, verifique se l1 e l2 são iguais. As listas l1 e l2 devem permanecer inalteradas. Esta função retorna 1 se as listas são iguais e 0, caso contrário. O protótipo desta função é o seguinte: **int igual (TLista *l1, TLista *l2).**
OBS: protótipo de função que pode ser útil: int strcmp (char *s, char *t).

Q9) Considerando a definição de lista de Q1, escreva uma função em C que, dadas duas listas, faça a concatenação das mesmas ao final de l1. O protótipo da função é o seguinte: **TLSE* junta_listas (TLSE* l1, TLSE* l2).**

Q10) Considerando a seguinte declaração de uma lista encadeada:

```
typedef struct lista{
        int mat;
        char nome[81];
        float cr;
        struct lista *prox;
}TL;
```

Escreva uma função em C que, dadas duas listas l1 e l2 encadeadas, verifique se l1 é a inversão de l2. As listas l1 e l2 devem permanecer inalteradas. Esta função retorna 1 se as listas estão invertidas e 0, caso contrário. O protótipo desta função é o seguinte: **int Contrario (TL *l1, TL *l2).**

Q11) Considerando a declaração da Q1, escreva uma função em C que, dada uma lista **l** qualquer, ordene os elementos de **l** em uma outra lista de saída. Portanto, a lista de entrada não pode ser alterada. O protótipo da função desta função é o seguinte: **TLSE * ordena (TLSE* l).**