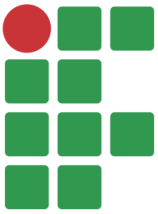

INSTITUTO FEDERAL DA PARAÍBA
CAMPUS CAMPINA GRANDE
BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO
DISCIPLINA DE LABORATÓRIO DE ESTRUTURAS DE DADOS
PROF. VICTOR ANDRÉ PINHO DE OLIVEIRA

Lab. Estruturas de Dados

Atividade Prática 4 - Listas Lineares Sequenciais

Instruções

Responda às questões abaixo, desenvolvendo cada uma em um novo arquivo .cpp ou .c. Temos 4 questões, sendo que as questões 1 e 2 valem 1 ponto cada; e as questões 3 e 4, 2 pontos cada.

Questões

1. Considerando o código da Lista Linear Sequencial não Ordenada presente no material, crie uma função inserir_ini que permite inserir um elemento no início da lista.
2. Considerando o código da Lista Linear Sequencial não Ordenada presente no material, faça:
    a. modifique a função inserir de modo que ela não permita a inserção de um novo elemento caso ele já exista na Lista
    b. modifique a função remover de modo que ela remova todas as ocorrências do elemento a ser removido
3. Considerando o código da Lista Linear Sequencial Ordenada presente no material, modifique-o de forma a termos uma Lista Linear Sequencial Ordenada de string (ordem lexicográfica ou alfabética). Considere o tamanho máximo para string de 20 caracteres (não esqueça do nulo).
4. A forma como implementamos nossas Listas no material traz uma grande limitação: o nosso programa só pode manipular uma Lista por vez, e isso não é bom. Podemos resolver isso criando um registro LISTA, que contém as variáveis particulares necessárias para o controle de cada LISTA. Desse modo, basta adicionarmos um novo parâmetro às funções para que elas operem em cima da Lista passada como argumento. A partir do código abaixo, implemente uma versão melhorada de uma Lista Linear Sequencial Ordenada de inteiros.

```c
#include <stdio.h>
#include <stdlib.h>
```

```
typedef struct {
 unsigned MAX;
 int *arr, pos;
} LISTA;

void criar(LISTA *lst, int tam_MAX);
void apagar(LISTA *lst);

void inserir_ord(LISTA *lst, int elemento);
void remover(LISTA *lst, int elemento);
int buscar(LISTA *lst, int elemento);

int obter(LISTA *lst, int indice);
int tamanho(LISTA *lst);
void imprimir(LISTA *lst);
```

Note que, da forma como criamos o registro, cada LISTA terá seu próprio tamanho máximo. Ao invés do array, temos um ponteiro para inteiro. O array precisará ser alocado, conforme comentaremos abaixo.

Considere:

- a função **criar** vai alocar dinamicamente espaço para o campo arr do registro LISTA considerando o valor passado ao segundo parâmetro: tam_MAX.
- a função **apagar** deverá desalocar o espaço alocado pela função criar.
- as demais funções farão a mesma coisa conforme visto no material. Desta vez, no entanto, considerando o parâmetro LISTA *lst, que é passada como ponteiro para cada função.

Em essência, o código está praticamente pronto no material, à exceção das funções criar e remover. Você fará apenas as adequações necessárias para atender às novas especificações.