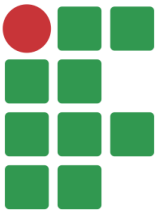

INSTITUTO FEDERAL DA PARAÍBA
CAMPUS CAMPINA GRANDE
BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO
DISCIPLINA DE LABORATÓRIO DE ESTRUTURAS DE DADOS
PROF. VICTOR ANDRÉ PINHO DE OLIVEIRA

Lab. Estruturas de Dados

Atividade Prática 5 - Listas Simplesmente Encadeadas

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 Simplesmente Encadeada não Ordenada presente no material, crie uma função inserir_ini que permite inserir um elemento no início da lista.
2. Crie uma Lista que armazena em cada nó o nome de um aluno, suas duas notas e a média. A Lista deve inserir os alunos já Ordenados pela média, de forma que a maior média deve está no início da lista e a menor no final.
3. 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 Simplesmente Encadeada de inteiros.

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

struct sNODE{
  int dado;
  struct sNODE *prox;
};

struct sLISTA{
  struct sNODE *ini, *fim;
};
```

```
typedef struct sLISTA LISTA;

void inicializar(LISTA *lst);
void apagar(LISTA *lst);

void inserir_ord(LISTA *lst, int dado);
void remover(LISTA *lst, int dado);
struct sNODE *buscar(LISTA *lst, int dado);

int obter(struct sNODE *node);
int tamanho(LISTA *lst);
void imprimir(LISTA *lst);
```

Note que, da forma como criamos o registro, cada LISTA terá seus próprios ponteiros ini e fim. Veja um exemplo de como criar e usar a LISTA:

```
int main(){
  LISTA lst;
  inicializar(&lst);

  inserir_ord(&lst, 100);
  imprimir(&lst);

  apagar(&lst);

  return 0;
}
```

Considere:

- a função **inicializar** apenas inicializa os ponteiros ini e fim para NULL.
- a função **apagar**, por sua vez, deverá desalocar todos os nós da lista. Não esqueça de atribuir NULL aos ponteiros ini e fim.
- 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 da função inicializar. Você fará apenas as adequações necessárias para atender às novas especificações.

4. Aproveitando o código implementado da questão anterior, crie uma nova função chamada juntar_ord que recebe duas Listas e gera uma nova Lista Ordenada contendo os elementos das duas Listas. Note que as Listas passadas como argumento se manterão inalteradas, você deverá alocar cada nó da nova Lista. A função deve seguir a seguinte assinatura:

```
LISTA juntar_ord(LISTA *lst1, LISTA *lst2);
```

Veja uma forma de como usar a função:

```
int main(){
  ...
  LISTA nova_lst = juntar_ord(&lst1, &lst2);

  imprimir(&nova_lst);

  apagar(&nova_lst);

  return 0;
}
```