

Lista de Exercícios 1 - LED - Laboratório de Estrutura de Dados

- 1 - Responda de forma breve qual a principal vantagem em adicionar o ponteiro fim em uma lista encadeada?
- 2 - Responda de forma breve qual seria a modificação necessária em uma lista encadeada com ponteiro no início e no fim para que seja possível implementar o caso de remoção no fim com um número de operações constante.
- 3 - Considere para resolução das questões o código a seguir:

Arquivo: mario.c

<pre>#include <stdio.h> #include <stdlib.h> typedef struct no{ char *nome; int num_vidas; struct no *prox; struct no *ant; }NO; NO *inicio = NULL; int tam = 0; void adicionar(char *nome, int num_vidas, int pos){ NO *novo = malloc (sizeof (NO)); novo->nome = nome; novo->num_vidas = num_vidas; novo->prox = NULL; novo->ant = NULL; if(tam == 0 && pos == 0){ inicio = novo; tam++; }else if(pos == 0){ novo->prox = inicio; inicio->ant = novo; inicio = novo; tam++; }else if(pos > 0 && pos <= tam){ //você deverá implementar este caso }else{ printf("insercao invalida! :/"); } }</pre>	<pre>void imprimir(){ //você deverá implementar essa função } void remover(char *nome){ if(tam == 1 && inicio->nome == nome){ NO *lixo = inicio; inicio = NULL; free(lixo); tam--; }else if(tam > 1 && inicio->nome == nome){ NO *lixo = inicio; inicio = inicio->prox; inicio->ant = NULL; free(lixo); tam--; }else { //você deverá implementar este caso } } char* persMaisVidas(_____)(){ //você deverá implementar essa função return nome; } int main(){ adicionar("Mario", 2, 0); adicionar("Luigi", 9, 0); adicionar("Princesa", 7, 0); adicionar("Toad", 3, 0); imprimir(); //você pode adicionar variáveis auxiliares caso ache necessário aqui. printf("Personagem c/ mais vidas: %s\n", persMaisVidas(_____)); return 0; }</pre>
---	--

- (a) **Explique e/ou desenhe ou então codifique e explique** como deveria ser implementado os casos das funções de inserção e remoção que estão faltando no código.
 - (b) Considerando a análise de complexidade simplificada vista em sala de aula (**linear ou constante**). Apresente a complexidade de cada um dos **casos das funções de inserção e remoção** do código acima, justifique de forma curta cada uma delas.
 - (c) **Explique e codifique** como deveria ser implementada a função **imprimir**.
 - (d) **Explique e codifique** a função **persMaisVidas** que deve retornar o nome do personagem que possui mais vidas da lista (considere para esta questão que não existem personagens com número de vidas iguais).
- 4 - Considerando agora que nem uma lista **encadeada** é aceito inserção de elementos iguais, modifique a lista **encadeada** para armazenar inteiros (positivos e negativos), e um atributo booleano que é verdadeiro se o número inteiro armazenado é positivo; e falso se ele é negativo.



- (a) Refaça as funções de adicionar e remover nesta lista.
- (b) Qual a complexidade de cada caso de cada função.
- (c) Crie um método que busca um elemento na lista pelo seu valor inteiro e não pela sua posição, retornando o atributo booleano deste elemento.
- (d) Qual a complexidade deste método? Por que?
- (e) Crie um método que remova todos os elementos negativos desta lista.

5 - Considere agora que na lista **encadeada não** serão aceitos números inteiros repetidos.

- (a) Refaça as funções a questão 3.1 efetuando as modificações necessárias.
- (b) Neste caso a complexidade de cada método muda? Qual/Quais? Por que?

6.1 – Dada uma lista **duplamente encadeada** com ponteiros **início** e **fim**. Considere que esta lista contenha um atributo **Conta Bancária** que tem um nome (dono da conta), número da conta, senha (apenas dígitos inteiros) e um saldo.

- (a) Refaça as funções de adicionar e remover na lista efetuando as modificações necessárias.
- (b) Qual a complexidade de cada caso de cada função?

6.2 - Desenhe e explique como funciona passo a passo a **inserção** e a **remoção** do n -ésimo elemento desta lista **duplamente encadeada**.

7 - Considere o seguinte programa para simular o gerenciamento de contas de um banco (uma lista **duplamente encadeada** de contas bancárias).

(a) Crie uma função denominada **cadastrarConta** que recebe um nome, um número e uma senha, cria o objeto conta com saldo igual a zero; e adiciona de **forma ordenada** em uma lista **duplamente encadeada** que simula as contas de um banco (você deverá modificar a **lista duplamente encadeada**, criada na questão 7.1(a), para que tenha apenas um método adicionar ordenado).

(b) Crie uma função denominada **visualizar** que não possui parâmetros; e deve apresentar o número e saldo de todas as contas cadastradas até o momento.

(c) Crie uma função denominada **removerConta** que recebe o número de uma conta bancária; e remove a conta bancária da **lista duplamente encadeada mantendo-a ordenada** (você deverá modificar as **lista duplamente encadeada**, criada na questão 7.11(a), para que tenha apenas um método remover ordenado).

(d) Crie uma função denominada **depositar** que recebe o número de uma conta bancária, uma senha e um valor; e **credita** o valor informado ao saldo da conta bancária. Para isso, você deve pesquisar a conta bancária pelo número na lista duplamente encadeada que simula a lista de contas bancárias.

(e) Crie uma função denominada **sacar** que recebe o número de uma conta bancária, uma senha e um valor; e debita o valor informado ao saldo da conta bancária. Para isso, você deve pesquisar a conta bancária pelo número na lista duplamente encadeada que simula a lista de contas bancárias.

(f) Crie uma função denominada **valor** que recebe o número de uma conta bancária e uma senha; e informar o valor do saldo da conta bancária. Para isso, você deve pesquisar a conta bancária pelo número na lista duplamente encadeada que simula a lista de contas bancárias.