**P1 – Estrutura de Dados 1 – Elisa**

1. (2 pontos) Considere uma lista dinâmica encadeada simples que contenha registros de alunos, ordenados em ordem crescente pelo número de matrícula (mat). Escreva uma função para inserir um novo registro aluno na posição correta da lista, de forma a mantê-la sempre ordenada. O protótipo da função é:

**int inserir\_lista\_ordenada(Lista \*li, Aluno \*aluno);**

**OBS:** A função recebe um ponteiro que aponta para o primeiro elemento da lista e o novo registro a ser inserido na lista. A função deve retornar se a operação foi possível ou não.

int inserir\_lista\_ordenada(Lista \*li, Aluno \*aluno)

{

if(li == NULL)

return 0;

Elemento \*no;

no = criar\_elemento();

no->aluno = \*aluno;

if( (\*li) == NULL ){

no->prox = NULL;

\*li = no;

}

else{

Elemento \*anterior, \*atual;

atual = \*li;

while(atual != NULL && atual->aluno.mat < \*aluno->mat){

anterior = atual;

atual = atual->prox;

}

if( atual == (\*li) ){

no->prox = (\*li);

\*li = no;

}

else{

no->prox = atual;

anterior->prox = no;

}

}

return 1;

}

1. (2 pontos) Escreva uma função que remova os *n* últimos elementos de uma lista dinâmica duplamente encadeada que contenha números inteiros. O protótipo da função deve ser:

**Int remover\_n\_ultimos(Lista \*li, int n);**

**OBS:** A função recebe como parâmetro um ponteiro li que aponta para o primeiro elemento da lista e o número *n* de elementos que serão removidos. A função deve retornar se a operação foi possível ou não.

int remover\_elemento\_fim(Lista \*li)

{

if (li == NULL) return 0;

if ((\*li) == NULL) printf("Lista vazia!");

else{

Elemento \*no, \*anterior;

no = (\*li);

do{

anterior = no;

no = no->prox;

} while(no->prox!=NULL);

if (no == (\*li))

(\*li) = no->prox;

else

anterior->prox = no->prox;

free(no);

}

return 1;

}

1. (2 pontos) Escreva uma função RECURSIVA para concatenar duas listas dinâmicas encadeadas simples. O protótipo da função deve ser:

**Int concatenar(Lista \*l1, Lista \*l2);**

**OBS:** A função recebe como parâmetro dois ponteiros: l1 e l2 que apontam para os primeiros elementos da primeira e segunda lista, respectivamente. A função deve retornar se a função foi possível ou não.

int concatenar (lista \*l1, lista \*l2)

{

if (l1 == NULL) return 0;

if (l2 == NULL) return 0;

if (l1->prox == NULL)

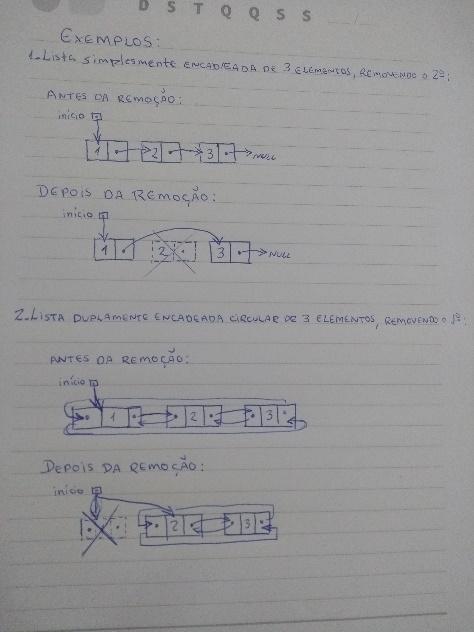
l2 = l1->prox;

no->prox = l1;

return concatenar(&no->prox;l2);

}

1. (1 ponto) Era para desenhar o que ocorre na memória antes e após a inserção dos elementos nas listas. Questões A, B, C, D. Exemplo:



1. Praticamente uma questão fechada sobre vantagens de desvantagens de uma lista dinâmica em C.

VANTAGENS

Melhor utilização dos recursos da memória.

Não é preciso definir previamente o tamanho da lista

Não é necessário movimentar os elementos nas operações de inserção e

remoção

DESVANTAGENS

Acesso indireto aos elementos

Necessidade de percorrer a lista para acessar determinado elemento.