Prova 1 – Recursividade e Listas Encadeadas

Instruções:

1. **Provas e/ou questões idênticas (ou com indícios de cópia) terão a nota zerada.**
2. A prova pode ser resolvida com consulta ao material impresso ou digital **off-line** (ou seja, não será permitido o acesso à internet durante a prova).
3. Ao término da prova, compacte a implementação de cada questão em um único arquivo (*com o seu nome completo*) e submeta no moodle na atividade denominada "**Prova 1 – Recursividade e Listas Encadeadas**".

Questão 1 *(2,5 pontos)* – Escreva um programa em C (recursivo) que calcule o MDC (Máximo Divisor Comum) de dois números inteiros.

int MDC (**int** a, **int** b)

$$\text{GCD(n,m)} = \begin{cases} GCD(n,m) & if \quad n < m \\ m & if \quad n \geq m \; and \; n \bmod m = 0 \\ GCD(m, n \bmod m) & otherwise \end{cases}$$

Por exemplo, para **a**=10 e **b**=50, o MDC é 10.

Questão 2 *(3,0 pontos)* **–** Escreva a função **slipt (**na biblioteca "*ListaEncadeada.h*") a qual divide uma lista **simplesmente** encadeada (recebida como parâmetro) em duas listas, conforme o critério dado pela função de predicado passada também como parâmetro.
A assinatura da função de predicado é dada a seguir:

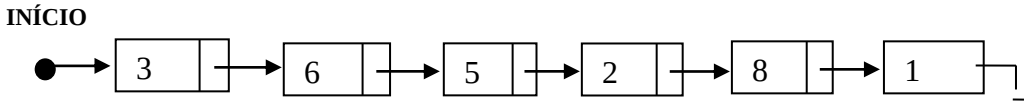
typedef int (*funcaoPredicado)(**void** *)

A função de predicado recebe um parâmetro do tipo "**void ***" e devolve um valor inteiro diferente de zero se o parâmetro tem o predicado desejado, caso contrário, retorna o valor zero.

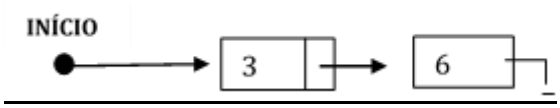
A assinatura da função **split** a ser implementada é dada a seguir:

struct** DLista* split (**struct** DLista *lista, funcaoPredicado fp**)

Exemplo: Considerando a lista encadeada a seguir, se a função de predicado dada verifica se o número inteiro é divisível por 3, então a função **split** gera como resultado uma nova lista contendo apenas os números divisíveis por 3 (ou seja, somente aqueles que satisfazem o predicado dado).



Para a lista do exemplo acima, a função **split** retornará a lista encadeada a seguir.



Questão 3 *(2,5 pontos)* **–** Escreva a função **moveMenor (**na biblioteca "*ListaEncadeada.h*") a qual encontra o menor elemento da lista e coloca-o como o primeiro da lista. O elemento da primeira posição ocupará o lugar do menor elemento (que foi movido pro início da lista).

void moveMenor **(struct** DLista *lista, FuncaoComparacao fc**)**

Questão 4 *(2,0 pontos)* – Escreva uma função **RECURSIVA** que encontra o maior elemento de uma lista simplesmente encadeada.

void* maiorElemento **(struct** DLista *lista, **void*** maiorAtual, FuncaoComparacao fc**)**