Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Medianeira

Disciplina: **Estrutura de Dados**

Prova 1 – Recursividade, Listas Lineares, Filas e Pilhas

Instruções:

1. **Provas e/ou questões idênticas (ou com indícios de cópia) terão a nota zerada.**
2. Para resolver as questões da prova, use obrigatoriamente as bibliotecas "*Lista.h*", "*Fila.h*" e "*Pilha.h*", conforme implementações e discussões realizadas em aula.
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, Listas Lineares, Filas e Pilhas**".

***Questão 1* (1,5 pontos)** – Escreva uma função recursiva MOD para calcular o resto da divisão de dois números inteiros positivos (x por y).

int MOD (**int** x, **int** y)

Por exemplo, o resultado de MOD(5,3) será 2.

***Questão 2* (2,5 pontos) –** Escreva a função **filtrar (**na biblioteca "*Lista.h*") a qual filtra as informações de uma lista encadeada (recebida como parâmetro) produzindo como resultado uma nova lista com as informações que atendem ao filtro dado (como parâmetro).

A lista original é alterada após a execução da função **filtrar**, sendo que as informações que atendem ao filtro são colocadas na lista a ser retornada e aquelas que não atendem ao filtro ficam na lista original.

A assinatura da função de filtro é dada a seguir:

typedef int (**FuncaoFiltro*)(**void** *)

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

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

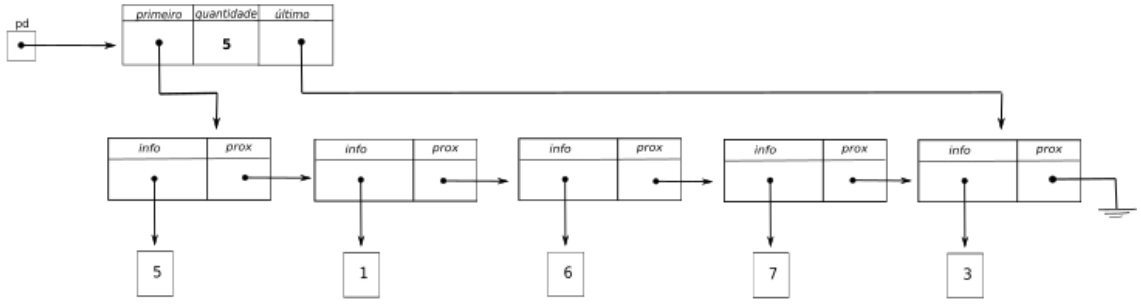
***pDLista** filtrar (**pDLista** lista, **FuncaoFiltro** ff)*

Exemplo: Considerando a lista encadeada de números inteiros a seguir, se a função de filtro verifica se um dado número é divisível por 3, então a função **filtrar** gera como

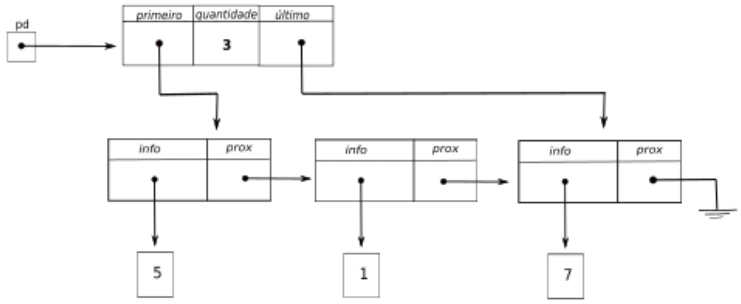
resultado uma nova lista contendo apenas aqueles números divisíveis por 3 (ou seja, somente aqueles que atendem ao filtro dado).

A figura a seguir ilustra o resultado produzido pela função **filtrar**.

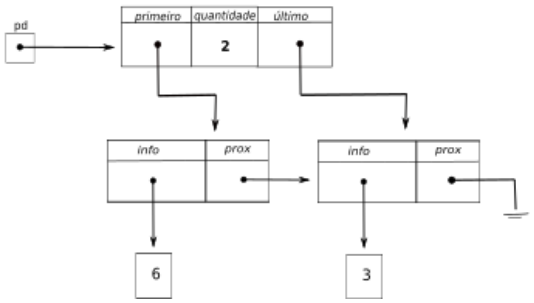
Lista Original



Lista 1 - lista original após a divisão



Lista 2 - lista com os elementos que satisfazem o predicado



Questão 3 - *(**1,5 pontos**)* – Escreva uma função **RECURSIVA** que encontra o maior elemento de uma lista encadeada.

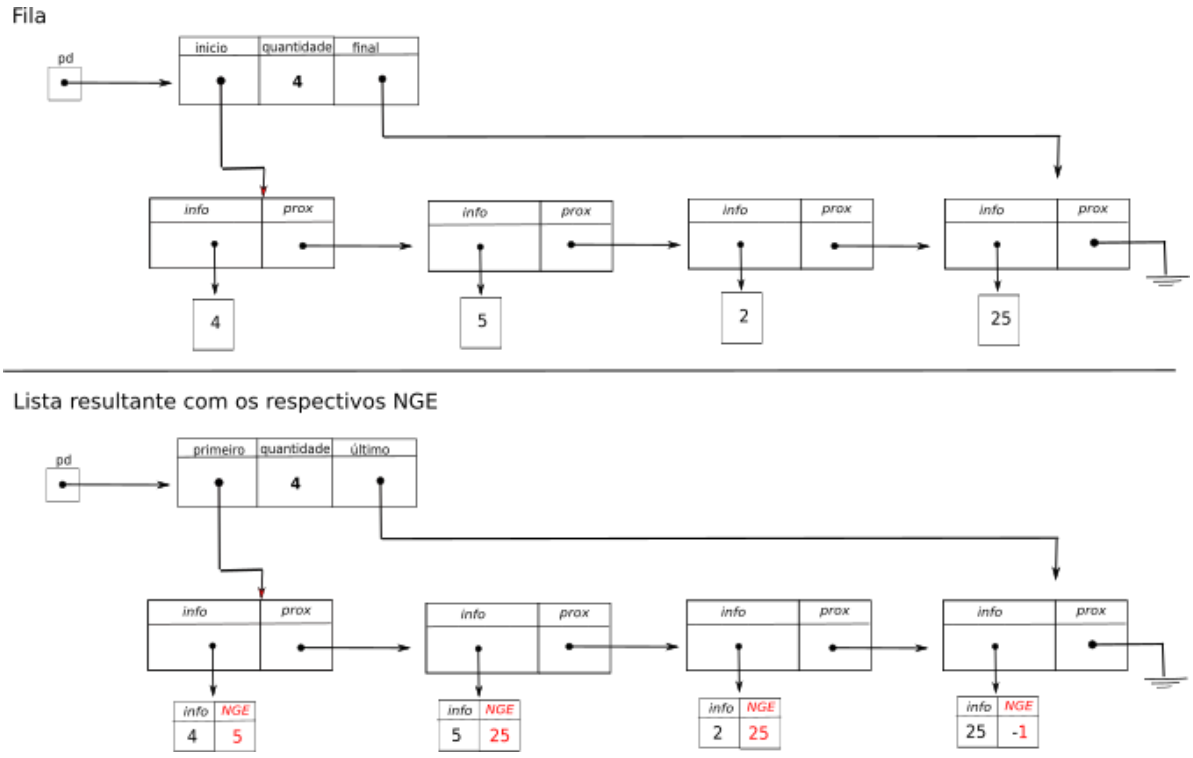
void* maiorElemento **(pDLista** lista, ***void**** maiorAtual, ***FuncaoComparacao*** fc**)**

Questão 4 – (2,5 pontos) Escreva uma função denominada **NGE (*Next Greater Element*)** que recebe como parâmetro uma ***fila*** e retorna uma ***lista encadeada*** onde cada elemento é um par que mapeia um elemento da fila com seu respectivo NGE (*veja o exemplo da figura a seguir*).

pDLista *NGE* (**pDFila** fila, **FuncaoComparacao** fc)

O NGE de um dado elemento ***n*** é o primeiro elemento maior que está na sequência da fila. Quando um elemento não tiver um NGE correspondente, considere o NGE sendo -1.

Exemplo: Para uma fila formada pela sequência de elementos [4, 5, 2, 25], o mapeamento dos elementos da fila com seus respectivos NGEs é ilustrado na figura a seguir.



Questão 5 **- *(2,0 pontos)*** Escreva uma função que determina se duas pilhas são iguais. Os valores **NÃO** precisam estar nas mesmas posições da pilha para que as pilhas sejam iguais, é suficiente que contenham exatamente os mesmos valores.

char iguais ***(pDPilha*** pilha1***, pDPilha*** *pilha2,* ***FuncaoComparacao*** *fc)*