# UNIVERSIDADE DA BEIRA INTERIOR

**Algoritmos e Estruturas de Dados** **2º Semestre**

**Frequência 1 (8 val)** **1:15 h + 10 min** **07/04/2025**

## A. [1.25 val] Análise de complexidade dos algoritmos

Considere o seguinte bloco de código em linguagem C:

```
...
for (k = 1; k <= n; k++) {
    for (j = k; j < k+2; j++)
        printf("OLA");
}
...
```

**1.** Simule o bloco de código dado, considerando que **n** é um número inteiro **muito grande**. Apresente os dados da simulação numa tabela cujo cabeçalho é o seguinte:

| k | k <= n (S/N) | j | j < k+2 (S/N) | printf |
|---|---|---|---|---|
| … | … | … | … | … |
| | | | | |

**2.** A partir dos resultados da simulação obtidos em 1, determine a **ordem de complexidade** do algoritmo associado ao bloco de código dado. Justifique, apresentando os cálculos efetuados.

## B. [1.50 val] Pilhas

Considere as seguintes declarações associadas a EAD Pilha:

```
typedef int INFOPilha;   // INFOPilha = int
struct NodoPilha { … };
typedef struct NodoPilha *PNodoPilha;
```

assim como os seguintes protótipos de funções já implementadas (operações básicas sobre a EAD Pilha):

```
PNodoPilha criarPilha ();
int pilhaVazia (PNodoPilha S);
PNodoPilha push (INFOPilha X, PNodoPilha S);
PNodoPilha pop (PNodoPilha S);
INFOPilha topo (PNodoPilha S);
```

Usando as operações básicas sobre uma EAD Pilha, implemente uma **função em C** que

- **receba** uma **pilha S** já preenchida com valores inteiros (parâmetro da função),
- **determine** e **devolva** o elemento (número inteiro) com valor **negativo** que se encontra **mais afastado** do topo da pilha **S** (ou seja, **mais próximo do fundo** da pilha), mas deixando a pilha S inalterada; se não há elementos negativos na pilha S, então devolver **0**.

Ex: S = [ -3, -12, 4, -5, -4, 5 ], com topo(S) = -3 => resultados: -4 e S = [ -3, -12, 4, -5, -4, 5 ]

## C. [1.75 val] Árvores Binárias (AB)

Considere as seguintes declarações associadas a EAD Árvore Binária:

```
typedef int INFOAB;   // INFOAB = int
struct NodoAB {
   INFOAB  Elemento;
   struct NodoAB  *Esquerda;
   struct NodoAB  *Direita;
};
typedef struct NodoAB *PNodoAB;
```

**Implemente uma função em C** que

- **receba** uma AB **T** e um número inteiro **N** (parâmetros da função),
- **determine** e **devolva** a quantidade de elementos de **T** com valor no campo **Elemento** maior que **N**.

## D. [3.50 val] Listas ligadas

Considere as seguintes declarações associadas a EAD Lista:

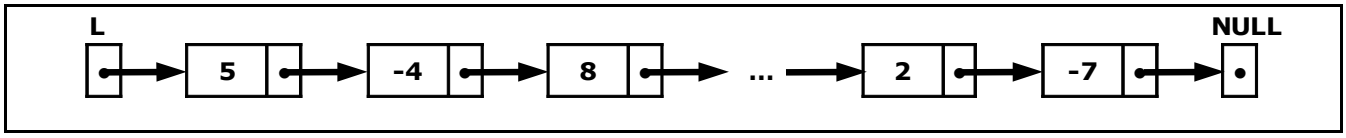
```
typedef int INFOLista;   // INFOLista = int
struct NodoLista {
   INFOLista  Elemento;
   struct NodoLista *Prox;
};
typedef struct NodoLista *PNodoLista;
```

assim como os seguintes protótipos de funções já implementadas (operações básicas sobre a EAD Lista):

```
int listaVazia (PNodoLista L); // devolve 1 se a lista L é vazia e 0 caso contrário
PNodoLista libertarNodoLista (PNodoLista P); // liberta a memória apontada por P e devolve NULL
```

Usando as definições e as funções dadas (protótipos), resolva as seguintes questões:

1. **[1.5 val] Implemente** uma **função recursiva em C** que
   - **receba** uma lista **L** de elementos do tipo **INFOLista** (inteiros) e um número inteiro **X** (parâmetros da função),
   - **determine** e **devolva** a soma dos valores do campo **Elemento** dos elementos de **L** com valores **menores** do que **X** (< X) naquele campo.
2. **[2.0 val] Usando** ou **não** as funções fornecidas em cima, mas **não podendo** usar outras funções, **implemente uma função iterativa (não recursiva) em C** que
   - **receba** uma lista **L** de elementos do tipo **INFOLista** (inteiros) (parâmetro da função),
   - **devolva** a lista **L** após **remover** o nodo com valor **positivo** no campo **Elemento** que se encontra **mais afastado** da cabeça de **L** (**mais próximo** da cauda), caso exista; se não existir, manter a lista L inalterada.

   **Nota:** Otimizar o algoritmo, de forma a que a lista L seja percorrida apenas uma vez.

   Exemplo: Na lista em baixo, o elemento a remover é o com o número **2**, pois é o elemento mais afastado da cabeça da lista com valor positivo.