IMD0029 - Estrutura de Dados Básicas 1 –2023.2 – Prova 02
Prof. Eiji Adachi M. Barbosa

Nome:_____

Matrícula: _____

ANTES DE COMEÇAR A PROVA, leia atentamente as seguintes instruções:

- Esta é uma prova escrita de caráter individual e sem consultas a pessoas ou material (impresso ou eletrônico).
- A prova vale 5,0 pontos na Unidade II e o valor de cada questão é informado no seu enunciado.
- Preze por respostas legíveis, bem organizadas e simples.
- As respostas devem ser fornecidas preferencialmente à caneta. Respostas fornecidas à lápis serão aceitas, mas eventuais questionamentos sobre a correção não serão aceitos.
- Celulares e outros dispositivos eletrônicos devem permanecer desligados durante toda a prova.
- **Desvios éticos ou de honestidade levarão a nota igual a zero na Unidade 2.**

Questão 1 (Responda na folha extra): (1,5 ponto) Considere uma lista duplamente encadeada com sentinelas cabeça e cauda, conforme a estrutura abaixo:

```
Lista {
   No* cabeca;
   No* cauda;
}
```

```
No{
   No* proximo;
   No* anterior;
   int valor;
}
```

Nesta questão, você deve implementar um método que verifica se a lista é simétrica ou não. Uma lista é dita simétrica se ela apresenta a mesma sequência de elementos quando lida no sentido da esquerda para direita quanto no sentido da direita para a esquerda. Por exemplo, a lista encadeada: 1 <-> 2 <-> 3 <-> 2 <-> 1 é simétrica porque possui a mesma sequência de elementos quando lida em ambos os sentidos, enquanto a lista 1 <-> 2 <-> 3 <-> 4 <-> 5 não é. A sua implementação deverá ter, obrigatoriamente, complexidade O(n) e deverá seguir a assinatura:

```
bool Lista::e_simetrica() – retorna true se é simétrica, false caso contrário.
```

Obs.: Você pode criar métodos auxiliares, caso ache necessário, desde que seu código também seja apresentado.

Questão 2: (1,5 ponto) Na computação, Conjunto é um Tipo Abstrato de Dados (TAD) que armazena uma coleção de elementos únicos, isto é, não há elementos repetidos num conjunto. As principais operações do TAD Conjunto são: (1) adicionar um elemento ao conjunto, (2) remover um elemento do conjunto e (3) checar se um determinado elemento está contido ou não no conjunto. Suponha que um Conjunto foi implementado em C++ usando a lista duplamente encadeada da questão 1. Considerando a operação de checar se um determinado elemento está contido no conjunto, e considerando sempre o pior caso dessa operação, existe vantagem em manter os elementos da lista sempre ordenados? Justifique sua resposta do ponto de vista da complexidade de tempo.

Questão 3 (Responda nos espaços abaixo de cada afirmação): (2,0 pontos) Para cada uma das afirmações a seguir, marque V (verdadeiro) ou F (falso), **justificando sucintamente** sua resposta. Marcações de V ou F **sem justificativas não serão aceitas.**

1 – ( ) Numa lista simplesmente encadeada com ponteiros para o início (primeiro nó) e para o fim (último nó) é possível realizar operações de inserir e remover no início e no fim da lista em complexidade O(1).

2 – ( ) Numa lista duplamente encadeada com sentinelas cabeça e cauda é possível realizar operações de inserir e remover no início e no fim da lista em complexidade O(1).

3 – ( ) Ao implementar uma estrutura de dados seguindo a estratégia *First-In First-Out* (FIFO) – primeiro a entrar primeiro a sair – usando um array, é possível implementar a operação de inserir com complexidade de tempo constante, mas é impossível implementar a operação de remover com complexidade de tempo constante.

4 – ( ) Ao implementar uma estrutura de dados seguindo a estratégia *First-In Last-Out* (FILO) – primeiro a entrar último a sair – usando uma lista simplesmente encadeada com ponteiro para o início, é impossível implementar as operações de inserir e remover com complexidade de tempo constante.