IMD0029 - Estrutura de Dados Básicas 1 – 2018.2 – Prova 01
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 I 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 em caneta. Respostas fornecidas a 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 1.**

Questão 1: (1,5 ponto) Sequências bitônicas inversas são aquelas que possuem duas sequências, sendo uma sequência inicial decrescente, seguida de uma sequência crescente. Ou seja, os elementos de uma sequência bitônica inversa obedecem a seguinte relação:

$$A_0 > A_1 > ... > A_{i-1} > A_i < A_{i+1} < ... < A_n$$

Considere que um array bitônico-inverso é um array de inteiros sem repetições cujos elementos representam uma sequência bitônica inversa. Neste contexto, implemente uma função que recebe como entrada um array bitônico-inverso e retorna o índice do elemento da "base". O elemento da "base" é o último elemento da sequência inicial decrescente e o primeiro elemento da sequência final crescente, ou seja, é o elemento Ai da relação acima. Sua função deverá obrigatoriamente ser **recursiva**, ter complexidade **$\Theta(\lg(n))$** e seguir a assinatura:

```
int findBase(int a[], int arraySize)
```

Obs.: Nesta questão, não podem ser usadas instruções para realizar repetição, como for, while e do-while. Ou seja, você não poderá usar instruções de repetição; você deverá construir sua solução apenas com chamadas recursivas.

Questão 2: (1,0 ponto) Dado um número natural N, implemente uma função **recursiva** que retorne a soma dos dígitos de N. Por exemplo, se N for igual a 2117, sua função deve retornar 11, que é o resultado da soma 2 + 1 + 1 + 7. Sua função deverá seguir a assinatura:

```
int sumDigit(int N)
```

Questão 3: (1,0 ponto) Explique em quais ocasiões o algoritmo de ordenação Quick Sort cai em seu pior caso, deixando claro qual a sua complexidade assintótica neste caso. Em seguida, explique uma estratégia de implementação que garante que o Quick Sort não cai no seu pior caso.

Questão 4: (1,5 ponto) 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 – ( ) Os algoritmos de busca sequencial e de busca binária podem ser empregados nas mesmas configurações de sequências de elementos passados como entrada.

2 – ( ) Os algoritmos de busca sequencial e de busca binária possuem mesma complexidade assintótica para o pior caso.

3 – ( ) Os algoritmos de ordenação Bubble Sort, Insertion Sort e Selection Sort possuem a mesma complexidade assintótica para o pior caso.

4 – ( ) Os algoritmos de ordenação Bubble Sort, Insertion Sort e Selection Sort possuem a mesma complexidade assintótica para o melhor caso.

5 – ( ) Os algoritmos de ordenação Quick Sort e Merge Sort possuem a mesma complexidade assintótica para o pior caso.

6 – ( ) Os algoritmos de ordenação Quick Sort e Merge Sort possuem a mesma complexidade assintótica para o melhor caso.

7 – ( ) O algoritmo de ordenação Merge Sort implementa uma estratégia de divisão e conquista recursiva em que a cada nível da recursão o problema de tamanho N é sempre dividido em dois sub-problemas de tamanhos N/2 (ou aproximadamente N/2).

8 – ( ) O algoritmo de ordenação Quick Sort implementa uma estratégia de divisão e conquista recursiva em que a cada nível da recursão o problema de tamanho N é sempre dividido em dois sub-problemas de tamanhos N/2 (ou aproximadamente N/2).