IMD0029 - Estrutura de Dados Básicas 1

Avaliação Prática em Laboratório – Unidade 1 - 2023.1

Prof. Eiji Adachi M. Barbosa

LEIA ANTES DE COMEÇAR

- Esta é uma atividade de caráter individual e sem consultas a pessoas ou material (impresso ou eletrônico).
- A atividade vale 5,0 pontos na 1a unidade. O valor de cada questão é informado no seu enunciado.
- Celulares e outros dispositivos eletrônicos devem permanecer desligados durante toda a prova.
- Desvios éticos ou de honestidade levarão à anulação da atividade do candidato (nota igual a zero).
- Junto a este enunciado, você também recebeu uma estrutura de diretórios contendo um diretório para cada questão. Em cada um destes diretórios, já existe uma assinatura de função e uma função `main` com um pequeno teste executável. A solução da sua questão deverá seguir a assinatura da função já estabelecida. Ou seja, não mude esta assinatura. Se necessário, crie funções auxiliares com outras assinaturas, mas não mude a assinatura da função original.

CRITÉRIOS DE CORREÇÃO

Para a correção desta atividade, serão levados em consideração, dentre outros, os seguintes pontos:

- Obediência às regras definidas para as assinaturas de função e para o entregável (arquivo .zip e estrutura de diretórios), conforme especificado no enunciado desta atividade.
- Existência de erros ou warnings de compilação do código fonte. Compile usando as flags `–Wall –pedantic –std=c++11`.
- Programas executam sem apresentar falhas e produzem os resultados esperados.
- Soluções atendem critérios de complexidade, caso estabelecido no enunciado.
- Apresentação e organização do código fonte entregue (identação, nome das variáveis, modularização do código em funções, etc).

Obs.: Para cada questão, já há uma função `main` com um pequeno teste executável. Este é um teste simples que não garante a corretude da sua implementação. Ou seja, se sua implementação passou no teste executável disponibilizado junto a este enunciado, isto não é garantia de que ela está totalmente correta. Para fins de correção, eu utilizarei outra bateria de testes mais completa, além de analisar manualmente o código produzido.

ENTREGÁVEL

O entregável desta atividade deverá seguir a mesma estrutura de diretórios do código fonte que você recebeu com este enunciado, obviamente, contendo os arquivos fonte utilizados para construir sua solução nos diretórios de cada questão. Além disso, o diretório pai deverá ter o seu nome e matrícula, seguindo o padrão `<PRIMEIRO_NOME>_<SOBRENOME>`

Dica: basta renomear o diretório /src com o padrão definido anteriormente.

Por exemplo:

```
> JOAO_SILVA
>   q1
>   q2
>   q3
>   q4
```

Toda esta estrutura de diretórios, incluindo os arquivos fonte com sua solução, deverá ser compactada num arquivo .zip (não é .rar, nem .tar.gz, nem qualquer outra extensão) que também deverá seguir o padrão `<PRIMEIRO_NOME>_<SOBRENOME>.zip`

O .zip não deve conter qualquer arquivo executável.

Este arquivo compactado deverá ser entregue via SIGAA até as 20:25. Este é um prazo fixo que não será estendido, exceto em casos muito excepcionais (ex.: SIGAA fora do ar). Ou seja, entregas após este horário não serão aceitas. A atividade do SIGAA permite apenas um envio, portanto certifique-se de que está enviando a versão correta antes de anexar ao SIGAA.

Questão 1 (1,5 ponto)

Nesta questão, implemente a seguinte função

```
int acharMenorOuIgual(int array[], int esquerda, int direita, int chave)
```

Esta função recebe um array de inteiros e um inteiro `chave` e retorna um inteiro que indica o índice do primeiro elemento que seja menor ou igual ao valor de `chave`, ou -1 caso não exista qualquer elemento menor do que `chave` no array de entrada. Os parâmetros `esquerda` e `direita` indicam os índices entre os quais está se realizando a busca.

Sua função deverá obrigatoriamente ser recursiva e ter complexidade $\Theta(\lg(n))$.

Obs.1: Assuma que o array já está ordenado em ordem crescente e que ele não possui elementos repetidos.
Obs.2: Nesta questão, não podem ser usadas instruções para realizar repetição, como for, while e do-while. Ou seja, você deverá construir sua solução apenas com chamadas recursivas.

Questão 2 (1,0 ponto)

Implemente a seguinte função:

```
int contarImpares(int array, int tamanho)
```

A função `contarImpares` deverá contar e retornar quantos números ímpares um array de entrada contém. Sua função deverá obrigatoriamente ser recursiva.

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

Questão 3 (1,0 ponto)

Implemente a seguinte função:

```
bool existeSoma3(int array[], int tamanho, int soma)
```

A função `existeSoma3` recebe como parâmetros um array de inteiros, o tamanho desse array e um valor `soma`. Essa função retorna `true` caso existam inteiros `i, j, k` tais que `array[i] + array[j] + array[k] == soma` (admite-se a possibilidade de `i, j, k` serem iguais); ou retorna `false`, em caso contrário. Sua solução deverá ter obrigatoriamente complexidade igual a $\Theta(n^2 * \log_2(n))$.

Obs.: Assuma nesta questão que o array de entrada sempre estará ordenado em ordem crescente.

Questão 4 (1,5 ponto)

Implemente o algoritmo de ordenação `QuickSort` usando a estratégia de seleção de pivô mediana de 3.

Obs.: Ao executar o arquivo executável, certifique-se de que o arquivo de entrada `input.txt` está no mesmo diretório que o arquivo executável.