# IMD0029 - Estruturas de Dados 1 - 2024.1 - Unidade 1

# Prof. Eiji Adachi

## LEIA ANTES DE COMEÇAR

- Esta é uma atividade de caráter individual e sem consultas a pessoas ou material (impresso ou eletrônico).
- 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` que implementa um teste executável simples; para rodá-lo, basta construir um executável a partir do arquivo `main.cpp` e executá-lo a partir da linha de comando.
- A solução da sua questão deverá seguir a assinatura da função já estabelecida, conforme o enunciado. 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.
- Existência de erros ou warnings de compilação do código fonte. Compile usando o `g++` ativando as opções `-Wall -Wextra`.
- 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 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 raiz deverá ter o seu nome, seguindo o padrão `<PRIMEIRO_NOME>_<SOBRENOME>`

Por exemplo:

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

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. Então remova os arquivos objeto e executáveis antes de compactar o seu entregável.

Este arquivo compactado deverá ser entregue via SIGAA até o horário estabelecido na tarefa. 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. Certifique-se de que está enviando a versão correta antes de anexar ao SIGAA.

## Questão 1 - Valor: 1.5

Dado um número natural N, implemente uma função recursiva que verifica se existe alguma sequência de pelo menos dois dígitos iguais e consecutivos em N, retornando verdadeiro em caso positivo e falso caso contrário. Por exemplo, se N for igual a 2113, sua função deve retornar `true`, pois existem dois dígitos iguais e consecutivos em N -- os 1's. Mas se N for igual a 2131, sua função deve retornar `false`, pois não existem dois dígitos iguais ocorrendo de modo consecutivo em N. Considere que se N possuir menos do que 2 dígitos a função deve retornar `false`.

Nesta questão, implemente a função seguindo a assinatura:

```
bool consecutivosIguais(int N)
```

A função deve ser implementada no arquivo `q1/main_q1.cpp`.

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 2 - Valor: 1.5

Implemente o algoritmo de ordenação Quick Sort de modo que os elementos do array de entrada estejam ordenados em ordem decrescente. Sua implementação deverá seguir a assinatura:

```
void quickSort(int array[], int esquerda, int direita)
```

A função deve ser implementada no arquivo `q2/main_q2.cpp`. Note que há nesse arquivo uma função auxiliar chamada `estaDecrescente` que eu criei para poder testar a ordenação. Você não pode alterar essa função auxiliar.

## Questão 3 - Valor: 2.0

Sequências bitônicas são aquelas que possuem duas sequências, sendo uma sequência inicial crescente, seguida de uma sequência decrescente. Ou seja, os elementos de uma sequência bitônica obedecem a seguinte relação:

A[0] < A[1] < ... < A[i-1] < A[i] > A[i+1] > ... > A[n]

Considere que um array bitônico é um array de inteiros sem repetições cujos elementos representam uma sequência bitônica. Neste contexto, implemente uma função que recebe como entrada um array bitônico e retorna o índice do elemento do "pico". O elemento do "pico" é o último elemento da sequência inicial crescente e o primeiro elemento da sequência final decrescente, ou seja, é o elemento A[i] da relação acima. Sua função deverá obrigatoriamente ser recursiva, ter complexidade Θ(lg(n)) e seguir a assinatura:

```
int acharPico(int array[], int esquerda, int direita)
```

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. A função deve ser implementada no arquivo `q3/main_q3.cpp`.