PROJETO FINAL –

PROF. DR. Alexsandro Monteiro Carneiro

1. DESCRIÇÃO

Este documento apresenta uma lista de projetos que cada grupo (até 3 alunos) vai desenvolver como Trabalhos Práticos (Atividade de Implementação), cujo peso na média final é de 30% presente nos instrumentos de avaliação desta disciplina. Este trabalho vai avaliar os itens abaixo, todos referentes a sua formação em algoritmos e programação:

1. Desenvolver a capacidade de compreender e propor algoritmos com qualidade e eficiência para a solução de problemas, independente de uma linguagem de programação.
2. Desenvolver a capacidade de codificar os algoritmos desenvolvidos em uma linguagem de programação.
3. Aprimorar a capacidade do aluno em inter-relacionar conteúdo das disciplinas já cursadas
4. Desenvolver a autonomia do estudante na busca por conhecimento e ferramentas. Utilizar os recursos e metodogias para compor projetos e expressar ideias.

2. AVALIAÇÃO

A **tabela -1** apresenta os critérios de avaliação a serem seguidos durante as entrevistas. Importante destacar que a avaliação é individual, semelhante as entrevistas anteriores.

ALUNO	Funciona (15%)	Domínio Técnico (50%)	Algoritmo Correto (10%)	Lógica (25%)

Tabela 1: critérios de avaliação

3. CALENDÁRIO DE ENTREVISTAS POR GRUPO

Cada grupo deve apresentar na data indicada (tabela 2) seguindo a sequência estabelecida. Os integrantes de cada grupo serão informados ao docente por meio do AVA. Os nomes dos integrantes de cada grupo serão inseridos na tabela 2.

⇨ **O número do grupo é o número do PROJETO (proposta)**

TURMA	DATA	GRUPO (Nº) - Alunos
A	12/06/2025	**Grupo 1 -** aluno 1: andré aluno 2: augusto aluno 3: abner
		Grupo 2 - aluno 1:Leonardo Modesto 196024 aluno 3:Lucas Petry 196183
		Grupo 3 - aluno 1: Lauro Gabriel 195915 aluno 2: Luan Pereira 183985 aluno 3: João Gabriel 192864
		Grupo 4 - aluno 1:João Lucas Souza 194461

		aluno 2:Pedro Stefanello 197226
		Grupo 5 - aluno 1:Thiago Carrato Alexandre 1929 aluno 2:Vitor Calixto Monzani 197343 aluno 3:Ian Luckemeyer Guimarães 196144
B	**17/06/2025**	**Grupo 6 -** aluno 1: Elyan Victor aluno 2: Jorge Gonçalves aluno 3: Diego Gabriel
		Grupo 7 - aluno 1:Gustavo 196724 aluno 2:Pedro 196925 aluno 3:Maria Rita 198554
		Grupo 8 - aluno 1:Nauã 198859 aluno 2:Breno 196331
		Grupo 9 - aluno 1: Gabriel Magno Senna 192951 aluno 2: Henrique Ramalho 191234 aluno 3: Raphael Antonio193081
		Grupo 10 - aluno 1: Diego Terra 198644 aluno 2: Filipe 199685

Tabela 2 – Lista de grupos e seus integrantes

LISTA DE PROJETOS

Para a avaliação foram definidos 10 projetos. Por meio de sorteio cada grupo vai obter um projeto e deve seguir exatamente as suas especificações.

Proposta 1: Gerenciador de Notas de Alunos (Selection Sort com Vetor de Registros)

Descrição: Desenvolver um sistema para gerenciar notas de alunos de uma turma. O programa deverá permitir cadastrar alunos (nome e notas de n disciplinas – Usuário define a quantidade. **Limitar em até 5 disciplinas**), listar todos os alunos com suas médias e ordenar a lista de alunos pela média final usando o algoritmo Selection Sort.

Requisitos:

- Modularização: Funções para cadastrar, listar e ordenar.
- Ponteiros: Uso de ponteiros para alocação dinâmica de um vetor de estruturas (registros) Aluno.
- Alocação Dinâmica: Alocar dinamicamente o vetor de alunos.
- Ordenação: Implementação do Selection Sort para ordenar os alunos pela média.
- Passagem de Parâmetros: A função de ordenação deve receber o vetor de alunos por referência.

Proposta 2: Análise de Temperaturas (Insertion Sort com Vetor Simples)

Descrição: Criar um programa que permita ao usuário inserir uma série de temperaturas diárias (alocação dinâmica- usuário define a quantidade. Programa não pode alterar este valor após sua definição, só em outra execução). O programa deverá calcular a média das temperaturas, identificar a maior e a menor temperatura registrada, e exibir as temperaturas ordenadas usando o algoritmo Insertion Sort**.**

Requisitos:

- Modularização: **Funções para inserção, cálculo de média/min/max e ordenação.**
- Ponteiros: **Uso de** ponteiros para alocação dinâmica **de um vetor de floats.**
- Alocação Dinâmica: **Alocar dinamicamente o vetor de temperaturas.**
- Ordenação: **Implementação do** Insertion Sort**.**
- Passagem de Parâmetros: **A função de ordenação deve receber o vetor por** referência

Proposta 3: Gerenciador de Estoque Simples (Merge Sort com Vetor de Registros)

Descrição: Desenvolver um programa para gerenciar um estoque simples de produtos. Cada produto terá um código, nome e quantidade. O programa deverá permitir adicionar produtos, listar o estoque e ordenar os produtos pelo código usando o algoritmo Merge Sort**. A quantidade de produtos é fixada em 10 produtos**

Requisitos:

- Modularização: **Funções para adicionar, listar e ordenar.**
- Ponteiros: **Uso de** ponteiros para alocação dinâmica **de um vetor de estruturas Produto.**
- Ordenação: **Implementação do** Merge Sort **(com função recursiva) para ordenar por código.**
- Recursividade: **A função mergeSort deve ser** recursiva**.**
- Passagem de Parâmetros: **A função mergeSort deve receber o vetor por** referência**.**

Proposta 4: Análise de Desempenho de Atletas (Selection Sort com Vetor de Registros)

Descrição: Desenvolver um programa para registrar e analisar o desempenho de atletas em uma modalidade esportiva (ex: corrida). O programa deverá permitir cadastrar atletas (nome, data: ou número inteiro que represente datas distintas e o tempo em segundos) e ordenar os atletas do menor tempo para o maior usando **Selection Sort**.

Requisitos:

- **Modularização:** Funções para cadastro, listagem e ordenação.
- **Ponteiros:** Uso de **ponteiros para alocação dinâmica** de um vetor de estruturas Atleta.
- **Alocação Dinâmica:** Alocar dinamicamente o vetor de atletas.
- **Ordenação:** Implementação do **Selection Sort** para ordenar pelo tempo.
- **Passagem de Parâmetros:** A função de ordenação deve receber o vetor por **referência**.

Proposta 5: Registro de Vendas Diárias (Insertion Sort com Vetor Simples)

Descrição: Criar um programa que registre as vendas diárias de uma loja por um período. O programa deverá calcular o total de vendas, a média diária e exibir as vendas em ordem crescente usando **Insertion Sort**.

Requisitos:

- **Modularização:** Funções para registro, cálculo e ordenação.
- **Ponteiros:** Uso de **ponteiros para alocação dinâmica** de um vetor de floats.
- **Alocação Dinâmica:** Alocar dinamicamente o vetor de vendas.
- **Ordenação:** Implementação do **Insertion Sort**.
- **Passagem de Parâmetros:** A função de ordenação deve receber o vetor por **referência**.

Proposta 6: Ranking de Filmes Assistidos (Merge Sort com Vetor de Registros)

Descrição: Criar um programa para registrar e classificar filmes assistidos. Cada filme terá um título e uma nota de 0 a 10. O programa deverá permitir adicionar filmes, listar o ranking e ordenar os filmes pela nota (do maior para o menor) usando **Merge Sort**.

Requisitos:

- **Modularização:** Funções para adicionar, listar e ordenar.
- **Ponteiros:** Uso de **ponteiros para alocação dinâmica** de um vetor de estruturas Filme.
- **Alocação Dinâmica:** Alocar dinamicamente o vetor de filmes.
- **Ordenação:** Implementação do **Merge Sort** (com função recursiva) para ordenar por nota (decrescente).
- **Recursividade:** A função mergeSort deve ser **recursiva**.
- **Passagem de Parâmetros:** A função mergeSort deve receber o vetor por **referência**.

Proposta 7: Classificador de Números Aleatórios (Selection Sort com Vetor Simples)

Descrição: Gerar um vetor de números inteiros aleatórios. O programa deverá exibir os números gerados e, em seguida, ordená-los em ordem crescente usando o algoritmo **Selection Sort**. O limite do tamanho do vetor pode ser 100 ou 1000 números, o usuário define este valor.

Requisitos:

- **Modularização:** Funções para geração, exibição e ordenação.
- **Ponteiros:** Uso de **ponteiros para alocação dinâmica** de um vetor de inteiros.
- **Alocação Dinâmica:** Alocar dinamicamente o vetor de números(opção 100 ou 1000).
- **Ordenação:** Implementação do **Selection Sort**.
- **Passagem de Parâmetros:** A função de ordenação deve receber o vetor por **referência**.
- **Passagem de Parâmetros por Valor:** A função para exibir o vetor deve receber o tamanho por **valor**.

Proposta 8: Gerador de Senhas (Insertion Sort com Vetor de Registros)

Descrição: Criar um programa que gere senhas (combinações de caracteres alfanuméricos) e as armazene junto com um identificador. O programa deverá permitir gerar senhas, listar as senhas geradas e ordená-las alfabeticamente pelo identificador usando **Insertion Sort**.

Requisitos:

- **Modularização:** Funções para geração, listagem e ordenação.
- **Ponteiros:** Uso de **ponteiros para alocação dinâmica** de um vetor de estruturas Senha.
- **Alocação Dinâmica:** Alocar dinamicamente o vetor de senha (até 20 senhas).
- **Ordenação:** Implementação do **Insertion Sort** para ordenar por identificador (string).

- **Passagem de Parâmetros:** A função de ordenação deve receber o vetor por **referência**.

Proposta 9: Análise de Notas de Disciplinas (Merge Sort com Vetor de Registros)

Descrição: Desenvolver um sistema para gerenciar notas de várias disciplinas. Cada disciplina terá um nome e a nota final. O programa deverá permitir cadastrar disciplinas, listar as disciplinas e ordenar a lista pela nota final (do menor para o maior) usando o algoritmo **Merge Sort**.

Requisitos:

- **Modularização:** Funções para cadastrar, listar e ordenar.
- **Ponteiros:** Uso de **ponteiros para alocação dinâmica** de um vetor de estruturas Disciplina.
- **Alocação Dinâmica:** Alocar dinamicamente o vetor de disciplinas.
- **Ordenação:** Implementação do **Merge Sort** (com função recursiva) para ordenar as disciplinas pela nota.
- **Recursividade:** A função mergeSort deve ser **recursiva**.
- **Passagem de Parâmetros:** A função mergeSort deve receber o vetor por **referência**.
- **Passagem de Parâmetros por Valor:** Funções para exibir a nota de uma disciplina individualmente podem receber a nota por **valor**.

Proposta 10: Processador de Dados Numéricos (Selection Sort com Vetor Simples)

Descrição: Criar um programa que permita ao usuário inserir uma sequência de números inteiros. O programa deverá calcular a soma e o produto dos números, e exibir os números ordenados em ordem decrescente usando o algoritmo **Selection Sort**.

Requisitos:

- **Modularização:** Funções para inserção, cálculo e ordenação.
- **Ponteiros:** Uso de **ponteiros para alocação dinâmica** de um vetor de inteiros.
- **Alocação Dinâmica:** Alocar dinamicamente o vetor de números.
- **Ordenação:** Implementação do **Selection Sort** para ordenar em ordem decrescente.
- **Passagem de Parâmetros:** A função de ordenação deve receber o vetor por **referência**.
- **Passagem de Parâmetros por Valor:** Funções para exibir a soma e o produto podem receber os valores por **valor**.

BOA SORTE!