IMD0029 - EDB1 - 2025.1 - Unidade 2

Prof. Eiji Adachi

📝 LEIA ANTES DE COMEÇAR

- Atividade individual sem consulta a pessoas ou materiais (impresso ou eletrônico).
- O valor de cada questão está indicado no enunciado.
- Mantenha celulares e outros eletrônicos desligados durante a prova.
- Desvios éticos resultarão em nota zero nesta unidade.
- Você recebeu diretórios para cada questão, cada um contendo um código base, um arquivo `Makefile` e um teste executável, que pode ser executado com `make run-test`.
  ⚠ O teste não garante a corretude completa da implementação.
- **Não altere a assinatura das funções fornecidas.** Você pode criar funções auxiliares, mas a assinatura principal deve ser mantida.

📌 Critérios de Correção

Serão avaliados os seguintes pontos:

- Conformidade com as **assinaturas de função** e estrutura de diretórios.
- **Compilação limpa**, sem erros ou *warnings* (use o `Makefile`).
- **Correta execução dos programas** com os resultados esperados.
- **Complexidade** conforme especificado no enunciado.
- **Qualidade do código**: organização, indentação, nomes adequados, modularização, etc.

📦 Entregável

- Use a mesma estrutura de diretórios recebida, com os arquivos de solução em cada pasta de questão.
- O diretório raiz deve conter seu nome em letras maiúsculas no formato `PRIMEIRO_NOME_SOBRENOME`. Exemplo:

```
JOAO_SILVA
├── q1
├── q2
└── q3
```

- Compacte tudo em um `.zip` com o mesmo nome: `PRIMEIRO_NOME_SOBRENOME.zip`.
- ❌ Não inclua arquivos `.o` ou executáveis, mas mantenha os arquivos `makefile`.
- Entregue via SIGAA até o horário estabelecido. Atrasos só serão aceitos com justificativa válida (ex.: instabilidade no SIGAA).

Questão 1 - Valor: 2.0

Implemente o método abaixo na classe `Lista`, que representa uma **lista duplamente encadeada com sentinelas cabeça e cauda**:

```
bool Lista::moverParaInicio(std::string str);
```

🔧 Comportamento esperado

- Recebe uma string como parâmetro e verifica se ela está presente em algum nó da lista.
- Se estiver, o primeiro nó correspondente deve ser desencadeado de sua posição atual (não deve ser deletado) e movido para o início da lista (não deve ser criado um novo nó), e o método retorna `true`.
- Caso contrário, a lista permanece inalterada e o método retorna `false`.

📌 Exemplo de uso

Lista original: `"S" <-> "P" <-> "F" <-> "C"`

Chamada do método	Estado esperado após execução	Retorno
`moverParaInicio("C")`	`"C" <-> "S" <-> "P" <-> "F"`	`true`
`moverParaInicio("E")`	`"S" <-> "P" <-> "F" <-> "C"`	`false`

📁 Onde implementar?

📄 Arquivo:

```
q1/src/Lista.cpp
```

Questão 2 - Valor: 2.0

Implemente o método abaixo na classe `Lista`, que representa uma **lista simplesmente encadeada com ponteiros para o primeiro e para o último nó da lista**:

```
int Lista::removerTodos(int val);
```

🔧 Comportamento esperado

- Recebe um valor inteiro `val` e remove **todos os nós da lista** com esse valor.
- A função deve retornar o número total de elementos removidos.
- ⏱️ Complexidade de tempo: **Θ(n)**
- ❌ Não usar estruturas auxiliares como `stack`, `array`, `vector`, etc.

📌 Exemplo de uso

Lista original: `{1 -> 7 -> 3 -> 7 -> 5 -> 7 -> 14}`

Chamada do método	Estado esperado após execução	Retorno
`removerTodos(7);`	`{1 -> 3 -> 5 -> 14}`	`3`
`removerTodos(20);`	`{1 -> 7 -> 3 -> 7 -> 5 -> 7 -> 14}`	`0`

📁 Onde implementar?

📄 Arquivo:

```
q2/src/Lista.cpp
```

Questão 3 - Valor: 2.0

Implemente os métodos abaixo na classe `Fila<T>`, que representa uma **fila genérica implementada com array**:

```
T frente(); // Retorna o primeiro elemento da fila (sem removê-lo)
void enfileirar(T elem);  // Insere um elemento no final da fila
T desenfileirar();  // Remove e retorna o primeiro elemento da fila
```

🔧 Comportamento esperado

- A fila deve seguir a estratégia **FIFO (First-In, First-Out)**.
- ⏱️ Complexidade de tempo (todas operações): **Θ(1)**

📌 Exemplo de uso

```
Fila<int> f(4);
f.enfileirar(10);
f.enfileirar(20);
f.enfileirar(30);
f.desenfileirar(); // remove 10
f.enfileirar(40);
f.enfileirar(50);
```

Estado final da fila:

```
{ 20 30 40 50 }
```

📁 Onde implementar?

📄 Arquivo:

```
q3/header/Fila.h
```

Observação: Lembre-se de sempre usar o comando `make clean run-test`