IMD0029 - EDB1 - 2025.1 - Unidade 3

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
├── q2e3
```

- Compacte tudo em um `.zip` com o mesmo nome: `PRIMEIRO_NOME_SOBRENOME.zip`.
- ❌ Não inclua arquivos `.o` ou executáveis. Para excluí-los, basta executar `make clean`.
- 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 `TabelaHash`, que representa uma **tabela hash com tratamento de colisão por encadeamento externo**:

```
void TabelaHash::inserirOrdenado(std::string chave, std::string valor);
```

🔧 Comportamento esperado

- O método deve inserir o par `{chave, valor}` na tabela hash respeitando a seguinte regra: o novo elemento deve ser inserido no bucket correspondente mantendo a **ordem crescente da chave**.
- Não é permitido ordenar o bucket após a inserção — o nó deve ser inserido **diretamente na posição correta**.
- A inserção ordenada deve ser feita manipulando apenas os nós dos buckets; não é permitido usar outras estruturas de dados auxiliares, como arrays, vectors, maps, etc.

📌 Exemplo de uso

Considerando uma tabela com 5 buckets (`capacidade = 5`), e a função de espalhamento `chave % 5`:

```
TabelaHash t(5);
t.inserirOrdenado(7, "A");    // chave 7  vai para bucket 2
t.inserirOrdenado(12, "B");   // chave 12 vai para bucket 2
t.inserirOrdenado(2, "C");    // chave 2  vai para bucket 2
t.inserirOrdenado(7, "Z");    // como a chave 7 já está no bucket, o valor
é atualizado
```

Estado final do bucket 2:

```
[(2, "C") -> (7, "Z") -> (12, "B")]
```

Observação: o exemplo acima usa chaves do tipo `int` apenas para facilitar o entendimento; no código, o método inserirOrdenado recebe chaves do tipo `std::string`. Para o tipo `std::string`, é possível usar todos os operadores de comparação `==`, `<`, `>`, etc.

📂 Onde implementar?

📄 Arquivo:

```
q1/header/TabelaHash.h
```

🧪 Como testar?

⚙️ Comando makefile:

```
make run-test
```

Questão 2 - Valor: 2.0

Implemente o método abaixo na classe `TabelaHash`, que representa uma **tabela hash com endereçamento aberto usando sondagem linear**:

```
bool TabelaHash::inserir(std::string chave, std::string valor);
```

🔧 Comportamento esperado

- A função deve inserir o par `{chave, valor}` na tabela, respeitando a lógica de endereçamento aberto usando sondagem linear em caso de colisão.
- **A tabela não deve ser redimensionada automaticamente.**
- Caso a tabela esteja cheia, deve lançar exceção do tipo `std::overflow_error`. Para isso, basta usar a instrução `throw std::overflow_error("Tabela cheia.");`.

📁 Onde implementar?

📄 Arquivo:

```
q2e3/header/TabelaHash.h
```

🧪 Como testar?

⚙️ Comando makefile:

```
make run-test-inserir
```

Questão 3 - Valor: 2.0

Implemente o método abaixo na classe `TabelaHash`, que representa uma **tabela hash com endereçamento aberto usando sondagem linear**:

```
void TabelaHash::redimensionar(int novaCapacidade);
```

🔧 Comportamento esperado

- Deve redimensionar a tabela com a capacidade fornecida por `novaCapacidade`.
- Deve apontar apenas para o novo array, liberando o espaço de memória do array antigo.
- Todos os pares ocupados do array original devem ser reinseridos no array novo.
- Não deve inserir nenhum par novo além dos já presentes como ocupados no array original.

📁 Onde implementar?

📄 Arquivo:

```
q2e3/header/TabelaHash.h
```

🧪 Como testar?

⚙️ Comando makefile:

```
make run-test-redimensionar
```