**IMD0029 - Estrutura de Dados Básicas 1 – 2018.1**

**Prof. Eiji Adachi M. Barbosa**

**Atividade Avaliativa Prática em Laboratório – Tabela de Dispersão**

**ANTES DE COMEÇAR,** leia atentamente as seguintes instruções:

* Esta é uma atividade de caráter individual e sem consultas a pessoas ou material (impresso ou eletrônico).
* A atividade vale 5,0 pontos na 3a unidade, e 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 arquivos fontes para auxiliar na construção da sua solução. No arquivo main.cpp, já existe uma função main com testes executáveis. 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!**

**Questão 1 (0,5 ponto):** Implemente o construtor HashTable::HashTable(unsigned long size) no arquivo HashTable.cpp. Este construtor recebe como parâmetro o tamanho do array interno da tabela de dispersão. Dica: você deve obrigatoriamente inicializar as posições do array com nullptr.

**Questão 2 (2,0 ponto):** Implemente o método HashTable::put no arquivo HashTable.cpp. Este método implementa a funcionalidade de inserir elementos numa tabela de dispersão. O tratamento de colisão deve ser realizado usando o método do endereçamento aberto por sondagem linear. Não é necessário se preocupar com o redimensionamento da tabela.

**Questão 3 (2,0 ponto):** Implemente o método HashTable::remove no arquivo HashTable.cpp. Este método implementa a funcionalidade de remover elementos da tabela de dispersão. O tratamento de colisão deve ser realizado usando o método do endereçamento aberto por sondagem linear. Não é necessário se preocupar com o redimensionamento da tabela.

**Questão 4 (0,5 ponto):** Implemente o destrutor HashTable::~HashTable() no arquivo HashTable.cpp. Dica: após implementar este destrutor, use o Valgrind para verificar se a memória alocada para a tabela de dispersão foi liberada corretamente. Obs.: Não existem testes específicos para esta questão; o destrutor é testado indiretamente pelos testes das questões 1, 2 e 3.

**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>\_<SOBRENOME>-<MATRICULA>. Por exemplo:

> JOAO\_SILVA-200012345

* src

Toda esta estrutura de diretórios, incluindo os arquivos fonte com sua solução, deverá ser compactada num arquivo .zip que também deverá seguir o padrão <PRIMEIRO>\_<SOBRENOME>-<MATRICULA>. 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.

**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), conforme especificado no enunciado desta atividade
* Existência de erros de compilação do código fonte
* Existência de vazamento de memória
* 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 função, 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 é apenas uma evidência mínima de que ela está correta. Para fins de correção, eu utilizarei outros casos de testes mais completos, além de inspecionar manualmente o código produzido.