**IMD0029 - Estrutura de Dados Básicas 1 – 2019.2**

**Prof. Eiji Adachi M. Barbosa**

**Atividade Avaliativa Prática em Laboratório – Unidade III**

**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 (1,5 ponto):** O TAD Multi-Conjunto, assim como o TAD Conjunto, não admite a repetição de elementos. No entanto, um Multi-Conjunto mantém uma contagem de quantas vezes um determinado elemento foi inserido; esta contagem é chamada de ‘multiplicidade’. Por exemplo: o Multi-Conjunto A = {X, X, Y} possui os elementos X e Y, mas o elemento X possui multiplicidade igual a 2 e o elemento Y possui multiplicidade igual a 1. Já o Multi-Conjunto B = {X, X, Y, Y, X} também possui os elementos X e Y, mas o elemento X possui multiplicidade igual a 3 e o elemento Y possui multiplicidade igual a 2. Neste exemplo, apesar de possuírem os mesmo elementos X e Y, os Multi-Conjuntos A e B não são iguais, pois a multiplicidade de seus elementos não é igual.

Nesta questão, você possui à sua disposição um TAD Dicionário já implementado corretamente e que provê as seguintes operações:

Dictionary::Dictionary ( ) // Construtor

bool Dictionary::Put( K key, V value ) // Inserir

bool Dictionary::Remove( K key ) // Remover

bool Dictionary::Contains( K key ) // Verifica se o dicionário contém a key

V Dictionary::Get( K key ) // Retorna o valor associado a key. Caso key não exista no dicionário, uma exceção será lançada

Com base neste TAD Dicionário, implemente uma função que recebe como entrada um array de strings e retorna um dicionário que representa um multi-conjunto contendo cada string única do array e suas respectivas multiplicidades. Por exemplo, se a entrada for igual a S = {X, X, Y, X, Y, Z}, a saída deverá ser um dicionário igual a D = { (X, 3), (Y, 2), (Z, 1) }. Sua solução deverá ter a seguinte assinatura (a função já existe no arquivo main.cpp):

Dictionary<string, int>\* buildMultiSet(string array[], int arraySize);

Obs.: Se você invocar o método *get* e a chave passada como parâmetro não existir no dicionário, uma exceção será lançada. Evite estes casos usando o método *contains* para certificar-se que a chave está ou não no dicionário antes de invocar o método *get*.

**Questão 2 (1,5 ponto):** O problema 2-Sum é definido da seguinte forma. Dado um array A de entrada e um valor inteiro K, retorne verdadeiro se existem dois elementos x e y em A tais que x + y = K, ou falso, caso contrário. Neste questão, você possui o TAD Conjunto com as seguintes operações:

Set::Set ( ) // Construtor

bool Set::Insert( K k) // Inserir

bool Set::Remove( K key ) // Remover

bool Set::Contains( K key ) // Verifica o conjunto contém key

Reuse este TAD Conjunto para resolver o problema 2-sum em tempo Θ(n). Pode assumir que implementação do conjunto garante, sob qualquer condição, que as operações Insert, Remove, e Contains são feitas em tempo Θ(1). Sua solução deverá ter a seguinte assinatura (a função já existe no arquivo main.cpp):

bool twoSum(int array[], int arraySize, int k);

**Questão 3 (2,0 ponto):** Implemente o método HashTable::resize(const unsigned long) no arquivo HashTable.cpp. Este método deve fazer o reajuste do array interno da tabela conforme visto na aula sobre Tabela de Dispersão Com Redimensionamento Dinâmico.

**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.