1. Resumo: Você deverá fazer em C++ uma implementação do TAD dicionário, usando como estrutura de dados as árvores AVL ensinadas em sala. O professor submeterá o seu código a uma série de *testes automáticos*, por isso é essencial atender de forma exata aos requisitos a seguir. Cada aluno deverá, individualmente, combinar com o professor a data e o horário para apresentação do trabalho.

Importante: lembre que o objetivo da disciplina é cada aluno aprimorar suas habilidades de programação e aprender os algoritmos ensinados, e que o caminho indicado pelo professor para que isso aconteça passa por cada aluno escrever o seu próprio código. Portanto, por favor colabore com o bom andamento da disciplina, não copiando códigos da internet ou de outros alunos, e leve em consideração que o professor está à disposição para ajudar. Obrigado.

2. Introdução: As declarações das funções que você deve implementar estão no arquivo `avl.hpp` a seguir (observe que, diferentemente do plano inicial, este trabalho não envolve iteradores, nem classes com seus respectivos membros públicos e privados):

```
// ========================================================================
typedef double TC;  typedef float TV;

struct Noh {  TC chave;  TV valor;  Noh *esq, *dir, *pai;  int h;  };

struct DicAVL { Noh *raiz; };      // Nulo quando árvore vazia.

void inicializar (DicAVL &D);      // Inicializa D como uma árvore vazia.

Noh* inserir (DicAVL &D, TC c, TV v);  // Retorna um ponteiro para o novo
                                       // nó, ou nulo se erro de alocação

Noh* procurar (DicAVL &D, TC c);   // Retorna um ponteiro para o nó da
                                   // chave procurada, ou nulo se a chave
                                   // não estiver em D.

void remover (DicAVL &D, Noh *n);  // 'n' aponta para o nó a ser removido

void terminar (DicAVL &D);         // Desaloca os nós da árvore.
// ========================================================================
```

A sua tarefa é escrever as definições dessas funções num arquivo `avl.cpp`, que deve então ser apresentado ao professor para a realização de testes sobre o código.

3. Requisitos: Segue abaixo uma especificação do restante do trabalho. Em caso de dúvida, por favor entre em contato com o professor rapidamente.

(a) **Não inclua uma função `main` no arquivo `avl.cpp`**, pois o programa de testes já possui tal função. Naturalmente, você precisará escrever uma função `main` para realizar os seus próprios testes antes de submeter o código ao professor, mas nesse caso essa função deve ficar noutro arquivo, que deverá fazer um `#include "avl.hpp"` e ser compilado junto com o arquivo `avl.cpp` para a realização dos seus testes individuais.

(b) **É importante que o endereço do nó de uma chave nunca mude!** Isto é, se, quando uma chave c é inserida, a função `inserir` retorna um ponteiro p, então, enquanto a chave c estiver no dicionário, qualquer chamada à função `procurar` em busca da chave c deve retornar o mesmo ponteiro p. Em particular, isso implica que, durante a remoção de uma chave, se um nó n vai ser substituído pelo seu sucessor (ou predecessor) s, então o nó s deve ser realmente "transplantado" para o local de n na árvore, e não apenas ter seus dados copiados para o nó n (pois isso alteraria o endereço do nó da chave armazenada em s).

4. Versões deste Documento:

(a) A versão 2 introduziu a restrição de que o endereço do nó de uma chave não pode mudar.