UNIVERSIDADE FEDERAL DE PELOTAS
CENTRO DE DESENVOLVIMENTO TECNOLÓGICO
CURSOS DE CIÊNCIA E ENGENHARIA DE COMPUTAÇÃO
Disciplina de Estruturas de Dados II – Trabalho Prático 1 – Turma M1 - 2016/2
Prof.: Felipe de Souza Marques (felipem@inf.ufpel.edu.br)

Nome: _____

Implementação de uma Tabela Hash

O trabalho consiste na implementação de uma tabela hash e 4 tipos de tratamento de colisão: encadeamento (listas encadeadas), sondagem linear, sondagem quadrática e hash duplo.

Requisitos da Tabela Hash

Espera-se que a implementação da tabela hash seja genérica, para que a mesma tabela possa suportar os 4 tipos de tratamento de colisão. A implementação deve suportar o uso de rehash. A hash deve suportar strings como chaves. Todas palavras estarão codificadas em ASCII e terão menos de 100 caracteres. Palavras são compostas por letras e hífens e sempre iniciam por uma letra. As letras podem ser maiúsculas ou minúsculas.

Deverá ser implementado um método de geração de código hash a partir de strings. A turma será dividida em quatro grupos. Cada grupo implementará um método de geração de hash. Os 4 métodos são FNV, Shift-Add-Xor, Xor e Rotating, disponíveis em:

http://eternallyconfuzzled.com/tuts/algorithms/jsw_tut_hashing.aspx.

Especificações da hash:

- Tamanho inicial do array: 500
- Fator de carga: 0.75
- Função de dispersão: Mod. Deste modo, a hash primária é $h1(k) = h(k) \bmod m$, onde $h(k)$ será determinado pelo método que codificará a string k e m é o número máximo de chaves que a tabela pode armazenar.
- Hash secundária (no caso de Double Hashing): $h2(k) = 1 + (h(k) \bmod (m-1))$
- Novo tamanho do array, após rehash: 2 vezes o tamanho anterior.
- Quando usar rehash? Imediatamente após a inserção, se o fator de carga atual (items_na_hash/tamanho_array) for maior **ou igual** ao fator de carga máximo (0.75)
- Ordem de iteração no rehash: iterar o array partindo do início (índice 0). No caso do encadeamento, iterar cada lista encadeada também partindo do início.

A estrutura deve ser capaz de suportar vários milhares de palavras e não utilizar muita memória. Apenas as bibliotecas padrões podem ser utilizadas (**exceto a STL, que não deve ser utilizada**).

Seu programa deve ser implementado em C ou C++, em arquivo único. Deve-se **utilizar a entrada e saída padrões (não abrir arquivos através do programa)**. Ele deve ser compilável em ambiente Linux (Ubuntu 12.04 ou melhor). O modo de tratamento de colisões deve ser escolhido por passagem de parâmetro (utilizando *argv* e *argc*), conforme descrito a seguir:

- -encadeamento: para a solução com listas encadeadas;
- -linear: para a solução com sondagem linear;
- -quadratica: para a solução com sondagem quadrática;
- -hash_duplo: para a solução com hash duplo.

Avaliação e Teste da Hash

O trabalho deve possuir um programa principal que receberá um arquivo texto de entrada contendo uma série de operações que devem ser realizadas pela tabela hash. As operações são de inserção, busca e remoção. A saída do programa consiste em um arquivo texto com um log das operações realizadas. No final do arquivo deve constar o numero total de colisões. O programa deve imprimir na tela o tempo de inserção, busca e remoção, para cada tipo de tratamento de colisão.

O log de saída consiste em uma série de informações por operação. Para cada operação, deve constar uma linha, com as seguintes informações, na ordem apresentada, separadas por um espaço:

- Nome da operação (INSERT, GET ou DELETE)
- A chave, entre aspas
- O código hash da chave (*h(k)*)
- O índice do vetor mapeado do código hash (*h1(k)*)
- O índice em que a chave foi inserida, obtida, ou removida. No caso de falha da operação, deve constar o índice o qual a chave deveria constar na tabela para que houvesse sucesso.
- O número de colisões. No endereçamento aberto, tentar inserir um elemento já existente não conta como uma colisão. Obviamente, podem ter ocorrido colisões antes do elemento ser encontrado, e elas devem ser computadas. No encadeamento, se o elemento a ser inserido, obtido, ou deletado constar na primeira posição da lista encadeada, isso **não** conta como uma colisão. Apenas é computada **uma** colisão quando for necessário percorrer a lista além do primeiro elemento.
- Flag informando sucesso ou falha (SUCCESS, FAIL).

A figura abaixo mostra um exemplo da sintaxe de entrada e saída. A sintaxe deve ser obedecida rigorosamente. Os códigos hash do exemplo e os códigos mapeados são completamente fictícios. Os demais dados estão coerentes com o funcionamento esperado de uma hash de sondagem linear.

Entrada	Saída
INSERT “ola mundo” INSERT “oi” INSERT “ola” INSERT “ola” GET “o” GET “a” GET “oi” GET “ola” DELETE “o” DELETE “ola mundo”	INSERT “ola mundo” 3212 5 5 0 SUCCESS INSERT “oi” 1345 1 1 0 SUCCESS INSERT “ola” 6757 1 2 1 SUCCESS INSERT “ola” 6757 1 2 1 FAIL GET “o” 145 7 7 0 FAIL GET “a” 5443 1 3 2 FAIL GET “oi” 1345 1 1 0 SUCCESS GET “ola” 6757 1 2 1 SUCCESS DELETE “o” 145 7 7 0 FAIL DELETE “ola mundo” 3212 5 5 0 SUCCESS 5

Avaliação

Sua tabela hash será testada em um amplo conjunto de testes e avaliada segundo o tempo total de execução e a quantidade de memória utilizada (uso máximo ao longo da execução). O tempo e memória utilizados devem ser menores que um valor máximo aceitável, ambos decorrentes de uma implementação minimamente correta.

Este trabalho tem peso de 30% da nota do bimestre.

Envio

O nome do seu arquivo fonte deve seguir o padrão **NomeSobrenome.[c|cpp]**. Em adição ao código fonte, também deve ser enviado um arquivo NomeSobrenome.txt contendo uma ***lista de bugs conhecidos*** do programa. Bugs descobertos que não estão nesta lista terão desconto maior. O envio de ambos será feito pelo AVA da disciplina, em um arquivo compactado.

Código de Honra

O trabalho deve ser implementado na sua totalidade, sem uso de bibliotecas prontas (fora as bibliotecas padrões - STL **não** deve ser utilizada) ou código de outros (colegas ou não). O trabalho enviado deve representar um esforço honesto em resolver o problema - isto é, não é algo "pela metade", que não implementa funcionalidades essenciais. Violações a esta conduta serão penalizadas e o violador não só terá nota nula

neste trabalho, mas também não terá direito a enviar os próximos trabalhos, ficando portanto sem parte da nota da disciplina. Plágios estão sujeitos a sanções administrativas pelo colegiado do curso.

Data de entrega

24 de Outubro de 2016.

Cada dia de atraso reduzirá em 10% a nota máxima possível de ser obtida.