Universidade Federal de Minas Gerais **Algoritmos e Estruturas de Dados II**
Instituto de Ciências Exatas **1º Semestre de 2017 - Turma TN**
Departamento de Ciência da Computação **Prof. Luiz Chaimowicz**

Trabalho Prático 2 – Índice Remissivo

Valor: 10 pontos
Data de devolução: 06/07/2017

O objetivo desse trabalho é utilizar alguns dos métodos vistos em sala para a construção de um índice remissivo de um texto. Basicamente, você deverá criar um programa que leia um texto qualquer (arquivo no formato *.txt*) e imprima, em ordem alfabética, todas palavras com 3 ou mais caracteres e a linha na qual elas aparecem no texto. (Esse tipo de índice costuma ser chamado de "arquivo invertido" e é a base de várias aplicações de recuperação de texto na web. Nesse caso, ao invés de armazenar linhas o seu arquivo armazena referências para os documentos no qual a palavra chave aparece).

Por exemplo, para o texto:

```
Um, dois, tres, testando.
Testando tres vezes tres.
Um teste final.
```

A saída deve ser algo como:

```
dois        1
final       3
testando    1 2
teste       3
tres        1 2
vezes       2
```

A leitura do arquivo deverá desprezar espaços em branco e sinais de pontuação, que serão considerados separadores de palavras, com exceção do hífen (-) que deve ser um caractere válido. Além disso, a leitura deverá converter todas as letras maiúsculas em minúsculas. Você pode considerar que cada palavra contém no máximo 20 letras e que não haverá caracteres acentuados. No caso de uma palavra aparecer mais de uma vez na mesma linha, basta mostrar uma única entrada (ex. acima).

Você deverá implementar o seu trabalho usando 2 TADs diferentes:

Árvore de Pesquisa sem Balanceamento: cada nodo irá armazenar uma palavra e uma lista de números das linhas onde ela aparece (use o TAD Lista para isso). A medida que as palavras são lidas, o seu algoritmo deve pesquisar a árvore para ver se a palavra já está presente. Se estiver, adiciona o novo número de linha à lista dessa palavra. Se não estiver presente, cria um novo nodo na árvore e inicia a lista de linhas com esta.

Hash com Tratamento de Colisões linear: cada entrada na tabela irá armazenar uma palavra e uma lista de números das linhas onde ela aparece (use o TAD Lista para isso). A medida que as palavras são lidas, o seu algoritmo deve calcular a função hash de cada palavra e pesquisar a tabela para ver se a palavra já está presente. Se estiver, adiciona o novo número de linha à lista dessa palavra. Se não estiver presente, cria uma nova entrada na tabela e inicia a lista de linhas com esta. Use a função hash vista em sala para transformar uma palavra em um número e escolha um valor adequado para M.

Para executar seu programa, use parâmetros de linha de comando para fazer sua chamada. Por exemplo, se o seu programa chama-se **TP2** e você quiser montar o índice do arquivo texto.txt e gerar a saída no arquivo saída.txt usando a árvore (ou o hash) sua chamada deve ser:

```
> TP2 texto.txt saida.txt TREE | HASH
```

No arquivo de saída as palavras deverão estar em ordem alfabética, uma por linha, seguidas das linhas onde aparecem separadas por espaços, como mostrado da página anterior.

Os arquivos de teste serão colocados em breve no Moodle.

O que deve ser entregue:

- Código fonte do programa em C (bem identado e comentado) e dos TADs auxiliares implementados. Demais detalhes sobre a submissão serão informados posteriormente
- Documentação do trabalho. Entre outras coisas, a documentação deve conter:

1. Introdução: descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa.
2. Implementação: descrição sobre a implementação do programa. Deve ser detalhada a estrutura de dados utilizada (de preferência com diagramas ilustrativos), o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, compilador utilizado, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado.
3. Estudo de Complexidade: estudo da complexidade do tempo de execução dos procedimentos implementados e do programa como um todo (notação O).
4. Testes: descrição dos testes realizados e listagem da saída (não edite os resultados). Os arquivos de testes serão fornecidos posteriormente.
4. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação. **Na sua conclusão faça também uma pequena discussão comparando as vantagens e desvantagens observadas no uso dos diferentes TADs**
5. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso

Comentários Gerais:

1 Comece a fazer este trabalho logo, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar.

2 Clareza, indentação e comentários no programa também serão avaliados.

3 O trabalho é individual.

4 Consulte regularmente o Moodle, uma vez que eventuais correções e modificações nesse enunciado poderão ser postadas lá.

5 A submissão será feita pelo Moodle. Faça um zip ou similar com todos os arquivos.

6 Trabalhos copiados serão penalizados conforme anunciado.

7 Penalização por atraso: $(2^d - 1)$ pontos, onde d é o número de dias (úteis) de atraso.