Universidade Federal de Minas Gerais **Algoritmos e Estrutura de Dados II**
Instituto de Ciências Exatas **2º Semestre de 2012**
Departamento de Ciência da Computação

Trabalho Prático 2

Prazo de Entrega: 10/12/2012 (impreterivelmente)
Valor: 20 pontos

1- Objetivo

O objetivo deste trabalho é realizar uma análise do número de ocorrências das palavras usadas em um texto baseada em uma nova ordem lexicográfica.

2- O Analisador

O programa deve ser capaz de ler todas as palavras contidas num arquivo de entrada e relatar no arquivo de saída todas as palavras que ocorrem no texto, seguido do número de vezes que esta palavra foi encontrada.

As palavras unidas por hífen (“-”) devem ser consideradas como uma só durante a análise e suas partes não devem ser contabilizadas separadamente.

Palavras que sejam encontradas mais de uma vez, com diferenciação de caixa alta ou baixa, deve ser contabilizadas como a mesma palavra. No arquivo de saída, as palavras encontradas devem sempre estar em caixa baixa.
Ex.:
As palavras “TRABALHO” e “tRaBaLhO” devem contabilizar duas ocorrências da mesma palavra no arquivo de saída - “trabalho”.

O programa deve ser capaz de ignorar espaços em branco e elementos de pontuação (“,”, “.”, “!”, “?”, “:”, “;” e “_”).

3- Entrada

A entrada do programa será feita através de um arquivo de entrada, passado como parâmetro durante a execução utilizando a flag abaixo:

-[i|I][1] : arquivo de entrada com o texto a ser processado e a nova ordem lexicográfica a ser usada

O arquivo de entrada será composto por dois blocos de informação: um bloco que define a nova ordem lexicográfica e outro que conterá um texto não formatado, composto de ao menos

[1] Os parâmetros podem ser informados com letras maiúsculas ou minúsculas

um caractere, que deverá ser tratado pelo programa. O bloco de definição da nova ordem lexicográfica será sempre iniciado pelo identificador “ORDEM”[2] seguido de uma quebra de linha. Em seguida, será informado a nova ordem lexicográfica para os caracteres, separados por um ou mais espaçamentos, seguindo o formato abaixo, seguido de uma quebra de linha:

$(CARACTERE)_1$ $(CARACTERE)_2$ …... $(CARACTERE)_{26}$

CARACTERE: Representa um caractere dentro intervalo de caracteres A-Z[3].

O bloco com o texto será iniciado pelo identificador “TEXTO”[2], seguido de uma quebra de linha. Em seguida será informado um texto que poderá conter uma ou mais linhas, sendo que cada linha será composta de pelo menos um caractere.

4- Saída

A saída do programa deve ser direcionada a um arquivo de saída cujo caminho será fornecido através do flag abaixo:

-[o|O] : endereço do arquivo de saída

O arquivo de saída deverá imprimir todas as palavras distintas do arquivo de entrada, em formato minúsculo, seguindo as especificações descritas na seção 2, seguido do número de vezes que determinada palavra aparece no texto de entrada. A comparação de ocorrências de uma palavra no texto deve ser *case sensitive*.

A saída do analisador deve ser ordenada de forma ascendente baseado na ordem lexicográfica informada.

Cada palavra no arquivo de saída deve ser separado por uma quebra de linha.
A última linha do arquivo deve conter o identificador “FIM”[2], seguido de uma quebra de linha.

5- Exemplo de Entrada e Saída

Entrada	**Saída**
ORDEM Z Y X W V U T S R Q P O N M L K J I H G F E D C B A TEXTO Era uma vez UMA gata xadrez.	xadrez 1 vez 1 uma 2 gata 1 era 1 FIM

[2] *Case sensitive*

[3] *Não é Case sensitive, portanto para a nova ordem lexicográfica ‘a’ == ‘A’*

6- Premissas

1. Não existe ordem pré-definida da ocorrência do bloco da ordem lexicográfica e do texto no arquivo de entrada
2. Não existe ordem pré-definida da ocorrência dos parâmetros de entrada e saída
3. Os caracteres do bloco da ordem lexicográfica sempre será um dos caracteres do intervalo A-Z (incluindo “K”, “Y” e “W”).
4. O programa gerado deve ser capaz de executar no servidor do Pratico com suas configurações padrão (problemas de estouro de memória, etc, **são responsabilidades do aluno**)
5. A execução do programa não deve durar mais que 60 segundos.
6. Não existem limites para a quantidade de caracteres numa linha do arquivo de entrada
7. Não existem limites para a quantidade de linhas no arquivo de entrada
8. Palavras agrupadas por hífen não possuem espaços em branco[3]
   Ex.: “guarda-chuva” **CORRETO**
   “guarda- chuva” **INCORRETO**
9. Nenhuma palavra do texto de entrada será acentuada[3]
10. Não existirão aspas simples ou duplas no texto de entrada
11. A nova ordem lexicográfica será apenas para o intervalo de caracteres A-Z, os demais seguirão sua ordem padrão.
12. As palavras “ORDEM” e “TEXTO” (*case sensitive*) são reservadas e não serão usadas no texto a ser processado
13. Nenhum sinal de pontuação estará presente no texto, exceto aqueles descritos na seção 2

7- Documentação

A documentação do trabalho deve conter, ao máximo, **5 páginas**, incluindo elementos não-textuais como capa, sumário, figuras, tabelas e referências. A documentação deve ser organizada e mostrar claramente os seguintes tópicos:

- Descrição do problema: apresentar qual o problema a ser tratado
- Implementação: descrever o algoritmo utilizado bem como as principais estruturas de dados
- Análise de Complexidade: apresentar a análise de complexidade em função do tempo na notação Big-*O*
- Testes: apresentar o conjunto de testes que validam a solução
- Conclusão: apresentar os ganhos e dificuldades no trabalho de forma clara e detalhada
- Referências bibliográficas: referências utilizadas no desenvolvimento do trabalho

[3] Esta premissa não é válida para aqueles que implementarem a parte extra do trabalho, descrita na seção 8

A documentação deve conter a descrição do seu trabalho em termos funcionais, dando foco nos algoritmos, estruturas de dados e decisões de implementação importantes durante o desenvolvimento.

Evite a descrição literal do código-fonte na documentação do trabalho.

Dica: Sua documentação deve ser clara o suficiente para que uma pessoa (da área de Computação ou não) consiga ler, entender o problema tratado e como foi feita a solução.

8- Extra (2 pontos) - Obrigatório para CC, SI e MatComp

Para prevenir erros de entrada, seu trabalho deve ser capaz de corrigir eventuais problemas no arquivo de entrada, antes de realizar o processamento sobre o texto.

Os possíveis erros que seu programa deve corrigir são:

- Conversão de caracteres acentuados para seu respectivo par, sem acentos.
- União de palavras com hífen separadas por espaços em branco
    - Todas as palavras terminadas com hífen deverão ser unidas a palavra subsequente
- Remoção de caracteres de pontuação (“,”, “.”, “!”, “?”, “:”, “;” e “_”) do início e meio de palavras

Após corrigir os problemas encontrados, o processamento sobre o texto deve ser o mesmo descrito na seção 2.

9- Comentários Gerais

A especificação pode sofrer alterações caso surja necessidade. Fique atento às aulas e ao Moodle. É de suma importância que logo após o lançamento da especificacão, suas dúvidas sobre a mesma sejam postadas o mais rápido possível nos auxiliando a identificar requisitos não especificados corretamente.

Dê preferência ao uso de Linux. Seu trabalho deverá ser **compilável em Linux** para ser devidamente corrigido pelo sistema.

O trabalho deve ser implementado em C e **não deve ser usada nenhuma biblioteca fora do padrão ANSI-C,** ou seja, bibliotecas do Windows não devem ser utilizadas.

O trabalho é individual. Trabalhos sem relatório não serão corrigidos em hipótese alguma.

Caso haja suspeita de que seu trabalho foi copiado, comprado, doado, etc você será avaliado de acordo e poderá ser convidado ou se convidar a explicar-se sobre os fatores que geraram a suspeita.

Programe de forma organizada modularizando o código de forma adequada. Comente seu código. Procure dar nome significativo para suas variáveis.

O trabalho será entregue através do sistema de submissão de trabalhos práticos: http://aeds.dcc.ufmg.br, em um arquivo compactado (.zip) contendo os arquivos fonte (arquivo .c, o arquivo .h, o main.c). A documentação deve ser enviada em pdf, no link especial para sua submissão no sistema.