Métodos de Programação III
LESI + LMCC (3^{o} ano)

Trabalho Prático nº 1
(Expressões Regulares em Lex)

Ano lectivo 2002/2003

O trabalho deve ser realizado em grupo,
e deve ser entregue acompanhado dum relatório, na forma que será indicada pelos docentes,
até 25 de Outubro de 2002 ao fim da tarde

1 Objectivos e Organização

Este trabalho prático tem como **objectivos** levar os alunos, através da resolução de problemas reais, a:

- praticar a programação baseada em Sistemas de Produção, ou seja Sistemas de Regras Condição-Acção;
- utilizar a expressividade total das Expressões Regulares para definir padrões de textos (sequências de caracteres);
- utilizar o gerador de reconhecedores/tradutores baseados em autómatos reactivos Lex, aplicando-o a problemas de processamento de textos (filtros, simuladores, interpretadores de comandos, etc.)

Para o efeito, esta folha contém cinco enunciados, dos quais deverá resolver pelo menos um.
O **relatório** a elaborar em LaTeX deve ser claro, completo e muito bem organizado, servindo com objectividade para a avaliação do seu trabalho, mas sobretudo constituindo um bom elemento de estudo, quer para si, quer para colegas seus; além do respectivo enunciado, análise do problema, decisões e detalhes de implementação, deverá conter exemplos de utilização, isto é, listagens de resultados obtidos que demonstrem bem a forma como concretizou a tarefa que lhe foi colocada.

2 Enunciados

2.1 Processador de Abreviaturas

Deverá desenvolver um programa *expansor de abreviaturas* que sempre que encontre num texto fonte uma abreviatura —no quadro abaixo apresenta-se um conjunto mínimo— a substitua no texto de saída pelo respectivo termo (uma ou mais palavras) por extenso.
Terminado o processamento do texto fonte, o programa deve indicar qual foi a abreviatura mais usada (ou indicar o número de utilizações de cada uma).

Abreviatura	Expansão
\qq	qualquer
\qd	quando
\tq	tal e qual
\td	todo
\ts	todos
\qb	quanto baste
\tb	também
\cf	conforme
\ag	a gosto
\ie	isto é
\pe	por exemplo
\cqd	como queríamos demonstrar
//	mente (sufixo) in (prefixo)

Contudo o processador deve deixar o texto fonte como está, sem fazer qualquer expansão, quando aparece no início de uma linha o comando \deixa. Esse comando fica activo até ao fim do texto, ou até aparecer, também no início de uma linha, o comando \expande.

2.2 Pré-processador para XML

Neste trabalho pretende-se que use o Lex para construir um processador para textos anotados em XML que simplesmente remova todas as marcas encontradas no texto fonte, devolvendo todo o conteúdo original. O texto intercalado entre uma marca de abertura (*palavra* entre `<` e `>`) e uma marca de fecho (*palavra* entre `</` e `>`) deve ser escrito na saída entre aspas (`"`).
Terminado o processamento do texto fonte, o programa deve mostrar a lista de marcas usadas (as tais palavras especiais que aparecem entre os delimitadores referidos), indicando o número de utilizações de cada uma.

2.3 Processador de Mensagens

Neste trabalho pretende-se que use o Lex para construir um processador para mensagens (de correio electrónico, ou tele-móvel) que:

- crie uma lista (a imprimir no fim do processamento) com todos os números do texto com probabilidade de serem um número de telefone nacional (segundo a actual norma portuguesa, 9 digitos começados por 2, 91, 96, 93, 8 ou 7);
- crie outra lista (a imprimir no fim do processamento) com todos os números do texto com probabilidade de serem um preços em euros (qualquer valor numérico maior ou igual a 0 na parte inteira (eventualmente separados por virgulas), e com 2 dígitos obrigatórios na parte decimal(separada da parte inteira por ponto));
- conte os *nomes próprios* e os *acrónimos/siglas* que ocorrem (provavelmente) num dado texto. Para efeitos de contagem, consideram-se *nomes próprios* as palavras começadas por uma letra maiúscula e continuadas por letras minúsculas e consideram-se *acrónimos/siglas* as palavras formadas só por letras maiúsculas. Se achar que vale a pena, imprima os nomes próprios e acrónimos/siglas encontrados, ordenados alfabeticamente e sem repetições.

Assumindo que um parágrafo é definido por uma mudança de linha imediatamente a seguir a um caracter terminador de período ('.', '?', '!'), pretende-se ainda que o processador em causa conte os parágrafos de um texto e insira, no texto de saída, uma linha formada por 10 caracteres mais ('+') no fim de cada parágrafo.

2.4 Normalizador de moradas

A normalização de bases de dados de endereços de clientes é sempre um problema, pois existem várias pessoas que inserem e alteram os registos, e cada um usa a sua forma de escrever uma morada. Pretende-se normalizar uma base de dados de moradas, uniformizando a forma de escrever as mesmas. Defina e concretize um conjunto de regras que melhorem a forma das moradas; quanto mais regras forem impostas, melhor será o resultado. Como sugestão, indicam-se alguns dos problemas encontrados.

1. O qualificativo Rua, Avenida, Praça, Praceta, etc. Exemplos:

   ```
   R.Camões
   Rua Frei Bernardo
   Avenida de Roma
   AV.ROMA
   ```

2. A forma de escrever o número de polícia. Exemplos:

   ```
   Avenida da Liberdade Nº 111
   Rua Bartolomeu dos Mártires, 1520
   ```

3. A forma de escrever o número do andar e o qualificativo do andar. Exemplos:

```
Avenida da Liberdade Nº 111-5º Dto
Rua Bartolomeu dos Mártires, 1520-7E
```

4. A forma de escrever o Rés-do-chão. Exemplos:

```
Avenida da Liberdade Nº 111-RC-ESQ
Rua Bartolomeu dos Mártires, 1520-R/C Ft
```

Para teste do seu programa, recomendamos que use o seguinte ficheiro de moradas: `http://www.di.uminho.pt/~prh/moradas.txt.gz`.