Processamento de Linguagens e Compiladores
LCC (3º ano)

1º Teste

Data: 15 de Janeiro de 2015
Hora: 09:00

Dispõe de 2:00 horas para realizar este teste

1 Filtros de Texto (8v)

I) Supondo que vai usar o gerador Flex para construir automaticamente filtros de texto, resolva as alíneas seguintes:

a) Explique cuidadosamente o que faz a especificação Flex abaixo:

```
%{
   char * mem[26]={[0 ... 25]=""};
%}

%option noyywrap
%x  III

%%
#[a-z]=.*       { mem[yytext[1]-'a'] = strdup(yytext+3); }
#[a-z]          { printf("%s",mem[yytext[1]-'a']);       }

\{\{            { BEGIN III; }
<III>.|\n       {            }
<III>\}\}       { BEGIN 0;   }

.|\n            { ECHO;      }
%%

int main(int argc, char* argv[]){
  if(argc==2){
    yyin = fopen(argv[1],"r");
       yylex();
    return 0; }
  else       {
    fprintf(stderr,"Erro\n");
    return 1; }
}
```

e concretize a sua explicação, indicando qual o texto que seria produzido à saída ao ler o seguinte texto de entrada:

```
Introducao ao alfabeto
#p=pedro
{{treta e mais treta}}
   #abc=sao as 3 primeiras letras.
e continua
#c='c' de cenoura.
   #d=de dado.
#e = de egua.
Repetindo
#abc ;
 #c d-#d
e por fim #e {{acabou}}}} exit
```

b) Escreva a especificação de um filtro para processar um texto e

- encontrar e listar alfabeticamente e sem repetições todos os possíveis nomes de verbos (palavras terminadas em "ar", "er", ou "ir");
- substituir todas as palavras marcadas pela *tag* `!acr{}` pela respetiva variante com todas as letras em maiúsculas (retirando a dita marca);
- substituir cada comando `!ref{etiqueta}` pelo número da linha, escrito entre parêntesis curvos, onde essa `etiqueta` foi definida com o comando `!label{etiqueta}`. Se ainda não foi definida, deve escrever `?n?` em que 'n' é o número de ocorrência dessa incógnita.

c) Analise as ER (expressões regulares) abaixo, escritas na notação do Flex, e explique por palavras suas, com clareza, cada uma dando 2 elucidativos exemplos de frases que concordam com elas:

```
%%
^[A-Z0-9]+[:\-]" "+
"//".*\n
[Ee][Ss][Tt][aAeE]
%%
```

II) AWK

a) Considere a seguinte script GAWK:

```
#!/usr/bin/gawk -f
BEGIN  { RS="href=[\"']"; FS="[\"']"; }
NR > 1 { print $1}
```

Indique o que ela faz quando aplicada a um ficheiro HTML. Para ilustrar a sua resposta, escreva um pequeno exemplo HTML e a respetiva saída.

b) Escreve um "oneliner"gawk que conte o numero de palavras e o numero de linhas de um texto.

2 Expressões Regulares e Autómatos (1v)

Considere as seguintes ER:

$$e1 \;=\; a\; b^*\; c \;+\; b\; (c \;+\; d)^+\; b$$

Responda, então, às seguintes questões:

a) usando a respectiva *cadeia de derivação*, diga se a frase "`bcdb`"pertence à linguagem gerada por $e1$.

b) construa informalmente o Autómato Determinista equivalente a $e1$.

3 Desenho/especificação de uma Linguagem (3+1v)

Para apoio a uma direção de curso (DC), num determinado ano letivo, pretende-se uma linguagem que permita descrever os pedidos de defesa de tese que essa DC recebeu num determinado período de receção aberto para o efeito. Por cada pedido o aluno candidato deve identificar-se pelo nome e número académico, o nome e o departamento do docente orientador (e o mesmo no caso de existir um co-orientador), o arguente (descrito pelo nome e a sigla da instituição à qual pertence) e por fim uma lista de datas (ano-mes-dia) possíveis para a defesa indicando-se para cada uma o período possível: manhã ou tarde.

Neste contexto, responda às alíneas seguintes.

a) Escreva, em notação do Yacc, uma Gramática Independente de Contexto, *GIC*, que especifique a Linguagem pretendida.

b) Escreva em Flex a especificação do Analisador Léxico para a linguagem definida pela *GIC* acima.

4 Gramáticas, Parsing e Tradução (7v)

A gramática independente de contexto, *GIC*, abaixo escrita em *BNF*, define uma linguagem de domínio específico para descrição de um arquivo de memórias familiares (aqui reduzido por questões óbvias).
O Símbolo Inicial é `ArqFam`, os Símbolos Terminais são escritos só em minúsculas (terminais-variáveis) ou só em maiúsculas (palavras-reservadas) ou entre apostrofes (sinais-de-pontuação), e a string nula é denotada por `&`; os restantes (sempre começados por maiúsculas) serão os Símbolos Não-Terminais.

```
p0:  ArqFam      -->  NomeFam Docs '.' Fotos '.'
p1:  NomeFam     -->  ARQUIVO DOS string ':'
p2:  Docs        -->  &
p3:               |   Docs ';' Doc
p4:  Doc         -->  Ref Tipo Desc Filename
P5:  Ref         -->  id Titulo
p6:  Titulo      -->  TIT '=' str
p7:  Desc        -->  str
p8:  Fotos       -->  Foto RFotos
p9:  RFotos      -->  ';' Fotos
p10:              |   &
p11: Foto        -->  FOTO str ano Filename
p12: Filename    -->  str
p13: Tipo        -->  CERTIDAO
p14:              |   NBIOGRAF
```

Neste contexto e após analisar a *GIC* dada, responda às alíneas seguintes.

a) Após estender a *GIC* dada, construa o estado inicial do autómato LR(0) e os estados que dele saem.

b) Transforme a *GIC* dada numa **gramática tradutora**, *GT*, reconhecível pelo Yacc, para:

- calcular e imprimir o número total de documentos e de fotos arquivados.
- imprimir o título dos documentos que são Certidões.
- verificar se os identificadores de documentos são realmente únicos.

c) Mostre que a frase `ARQUIVO DOS "Almeidas":  .  FOTO "Casa do Cedro" 1880 "./Almeidas/Casas/f1.jpg".` pertence à linguagem, construindo a respectiva Árvore de Derivação.

d) Calcule o `lookahead()` de todas as produções.

e) Construa a *Tabela de Parsing LL(1)* que indica para cada símbolo NT e para cada símbolo T qual a produção a usar para continuar o reconhecimento (ou terminar com erro, se o terminal não for aceite nesse momento). No fim, indique se existem *Conflitos LL(1)*.

f) Escreva as funções de um parser RD-puro (recursivo-descendente) para reconhecer os Símbolos $Fotos$ e $RFotos$.