Processamento de Linguagens e Compiladores — LCC (3ºano)

Exame de Recurso

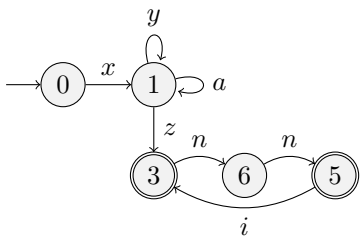
Data: 24 de Janeiro de 2018
Hora: 09:00

Dispõe de 2 horas para realizar este exame

Questão 1: Expressões Regulares e Autómatos (4v)

Responda, então, às seguintes questões:

a) Qual a expressão regular correspondente ao seguinte autómato:



b) Em língua portuguesa uma lista `[ana rui eva]` é expressa por *ana, rui e eva*. Escreva uma expressão regular que usando `grep -oP expreg` permita extrair todas as listas de duas ou mais palavras de 3 ou mais letras, contidas num texto.

Qual a saída de `grep -oP expreg` quando aplicada ao texto:

```
Título: Actividades gerais dos insectos e aranhas
Por: Sicrano, Beltrano e Fagundes
Os mosquitos picam gatos, salamandras, ratos e crocodilos.
Não picam, porém, os outros insectos, aves e peixes.
```

c) O comando `sed`, disponível em Linux, permite (entre outras coisas) fazer substituições de uma expressão regular, `expReg`, por uma `string` em todas as ocorrências desse padrão num dado ficheiro de texto `file`, sendo usado do seguinte modo

```
sed -re 's/expReg/string/g' file
```

Construa um comando `sed` que, usando uma única substituição esconda os autores de tum texto LaTeX (ex: `\author{Rui da Silva}`) substituindo-o por `\author{anonimous}`.

d) Desenhe um autómato determinístico correspondente a: $y(cb|aa)^{+}a$

Questão 2: Filtros de Texto em Flex e GAWK (4v)

Para ajudar a estudos sobre a língua portuguesa, foram disponibizados grandes quantidades de textos anotados linguisticamente, seguindo o sequinte formado:

```
<meta>
jornal publico
</meta>
<s>
Estes   este    DET 0   P   M   >N  0
temas   tema    N   0   00  0   SUBJ>   0
serão   ser V   FUT 3P  0   FS-STA  0
retomados   retomar V   PCP P   M   ICL-AUX<   0
na  em+o    PRP+DET   0   S   F   <ADVL+>N   0
próxima próximo ADJ 0   S   F   >N  0
reunião reunião N   0   00  0   P<  0
de  de  PRP   0   0   0   N<  0
de  de  PRP   0   0   0   N<  0
Câmara  câmara  N  0   00  0   P<  0
do  de+o    PRP   0   S   M   N< +>N  0
distrito    distrito    N  0   00  0   P<  0
de  de  PRP   0   0   0   N<  0
Leiria  Leiria  PROP    0   00  0   P<  0
,   ,   PU  0   0   0   PONT    0
a   a   PRP 0   0   0   <ADVL   0
realizar    realizar    V   INF 3   0   ICL-P<  0
no  em+o    PRP+DET   0   S   M   <ADVL+>N   0
dia dia N   0   00  0   P<  0
de  de  PRP   0   0   0   N<  0
Abril   abril   N 0   00  0   P<  0
,   ,   PU  0   0   0   PONT    0
nas em+o    PRP+DET   0   P   F   <ADVL+>N   0
Caldas  Caldas=da=Rainha    PROP    0   00  0   P<  0
da  Caldas=da=Rainha    PROP    0   00  0   P<  0
Rainha  Caldas=da=Rainha    PROP    0   00  0   P<  0
.   .   PU  0   0   0   PONT    0
</s>
<s>
...
```

Onde as colunas estão separadas por um tab (`\t`) e contêm respectivamente: palavra, lema, categoria gramatical, (ignore as outras colunas).

a) Escreva uma script *GAWK* que calcule quantas vezes ocorreu cada par (lema-categoria) num texto. Exemplo de saída esperada:

```
este-DET    83
tema-N      23
...
```

b) Desenvolva um Filtro recorrendo ao sistema *Flex* que reconstrua o texto original, imprimindo apenas o conteudo do *meta* e a primeira coluna das *s* de acordo com o seguinte formato:

```
==jornal publico
1. Estes temas serão retomados na ..... , nas Caldas da Rainha .
2. A câmara de .....
```

Questão 3: Desenho/especificação de uma Linguagem (4v=3+1)

A linguagem JSON é muito usada para intercâmbio de informação, ligada ao JavaScript, e presentemente há bibliotecas Json para quase todas as linguagens e ambientes. Genericamente expressa listas generalizadas, sendo as listas expressas entre `[ ... ]`, os mapeamentos `{ "a" : "b" , ... }`.
Considere o seguinte exemplo da linguagem

```
["a", {"bbb": "cc", "d": ["eee"]}]
```

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

b) Especifique em Flex um Analisador Léxico para reconhecer todos os símbolos terminais da sua linguagem e devolver os respetivos códigos.

Questão 4: Gramáticas, Tradução e Parsing Bottom-Up (8v)

Considere a seguinte gramática

```
registoParoquial →  registo registoParoquial
                 |

registo     → "#" data batismo
            | "#" data nascimento

batismo     → "batismo-de:" nome

nascimento → "nascimento:" nome "mãe:" nome "pai:" nome

nome → ID
```

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

a) Apresente uma frase *típica* desta linguagem e apresente também a frase mais curta.

b) Desenhe o autómato LR0 (apenas o estado inicial e mais 4) e indique se há conflitos LR0.

c) Diga porque é que esta gramática não é LL1 e escreva uma que, definindo exatamente a mesma linguagem, seja LL1.

d) Usando a estratégia de parsing recursivo descendente, escreva as funções para reconhecer o símbolo `registoParoquial`, `batismo` e qualquer símbolo terminal.

f) Se num dado momento do parsing a stack de parsing contiver os seguintes símbolos

```
data | batismo | registoParoquial | $
```

e o próximo símbolo da entrada a ser lido e aceite for `data`, diga justificando claramente qual é o método de parsing que está a ser usado?

g) Usando a notação do Yacc, estenda a GIC com ações semânticas para calcular o número de registos de batismo.

h) Usando a notação do Yacc, estenda a GIC com ações semânticas para verificar se todos os nascidos foram batisados (o nome será o identificador de cada pessoa).

i) Usando a notação do Yacc, estenda a GIC com ações semânticas para garantir que a mesma pessoa nunca é batisado antes de nascer.