Processamento de Linguagens – MiEI

Teste

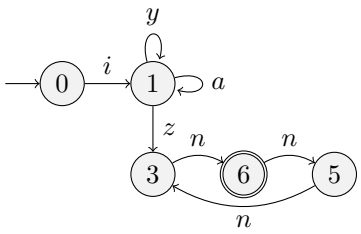
30 de Maio de 2018 (8h30)

Dispõe de **2:00 horas** para realizar este teste.

Questão 1: Expressões Regulares e Autómatos (3v)

Responda às seguintes alíneas:

a) Qual a expressão regular correspondente ao seguinte autómato:



b) Para ajudar a decorar o número da porta da namorada (nº 958), optou por procurar n'Os Lusíadas sequências de 3 palavras com comprimentos 9, 5, 8 obtendo: "valerosas obras exercita", "Ocidental praia Lusitana"(o que, para além de facilitar fixar o número, funciona também de facilitador de conversa).

Indique como é que usando `grep`, conseguiu fazer tal façanha! (Evite soluções que devolvam palavras parciais).

c) Considere as seguintes expressões regulares abaixo, escritas em notação do Flex:

```
1   (dia|DIA)
2   [dia|DIA]
3   [Dd][Ii][Aa]
4   (?i:dia)
5   dia/DIA
6   dia{1,3}
7   {dia}
```

Apresente todas as frases que cada uma delas *apanha*. No caso de serem muitas, apresente 6 das mais curtas. Comente quando necessário

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

Especifique filtros de texto com base em expressões regulares e regras de produção (padrão–ação) para resolver as seguintes alíneas:

a) *Matrículas de outro mundo*

Num país algures, as matrículas dos automóveis seguem os seguintes requisitos:

- Uma matrícula tem 8 dígitos;
- Os 8 dígitos estão divididos em 4 partes iguais de 2 dígitos por um separador que pode ser '...', '-' ou ':';
- Cada matrícula só pode ter uma espécie de separador;
- Os separadores têm de ter dígitos antes e depois, não há espaços.

Especifique um filtro em flex que apanhe e contabilize as matrículas num texto.

O texto contém ainda vários comentários multilinha em sintaxe C (`/*.....*/`) que deve ignorar.

b) Para pôr ordem na casa construiu-se uma listagem dos livros com o seguinte aspecto:

```
Volfrâmio : Aquilino : comprado 2009
Os Maias  : Eça : presente da avó Maria
Os Lusíadas : Camões : feira do livro 2011
Os Maias: Eça de Queirós : versão anotada
```

Constatou-se que há algumas repetições que pretende encontrar.

Escreva um filtro usando o gawk para anote títulos de livros duplicados de modo a obter:

```
Volfrâmio : Aquilino : comprado 2009
Os Maias  : Eça : presente da avó Maria
Os Lusíadas : Camões : feira do livro 2011
==2==Os Maias: Eça de Queirós : versão anotada
```

Notas:

(1) Nas linha contendo um título repetido, anotou-se o número da primeira ocorrência, de acordo com o exemplo.

(2) Não analise nem altere linhas com menos de 5 caracteres.

Questão 3: Desenho/especificação de uma Linguagem (7v=2+1+1+3)

Pretende-se criar uma aplicação para gerir listas de compras. Um dos seus componentes é uma linguagem com a qual se poderão descrever essas listas.

Neste contexto, responda às alíneas seguintes:

a) Especifique uma GIC (Gramática Independente do Contexto) para descrever listas de compras atendendo aos seguintes requisitos:

1. Uma lista de compras está dividida em secções em que cada secção corresponde a uma categoria de compra: limpeza, frescos, peixe, carne, padaria, ...
2. Cada secção tem, por sua vez, uma lista de produtos a comprar;
3. Cada produto é caraterizado por: código, designação, preço (de referência) e quantidade a comprar;
4. A quantidade a comprar é um tuplo formado por tipo (unidade, peso, volume líquido, ...) e valor.

Assegure-se que a sua GIC verifica a Condição LL(1).

b) Usando a gramática especificada em cima apresente um exemplo de uma lista de compras e construa a respectiva Árvore de Derivação.

c) Usando a notação do Flex especifique um analisador léxico para esta linguagem.

d) Usando notação do Yacc (e todas as facilidades oferecidas pelo par de ferramentas Flex/Yacc) transforme a sua GIC numa **gramática tradutora (*GT*)** (juntando-lhe ações semânticas) para:

d1) contar o número de produtos que vai comprar;

d2) usando os preços de referência, apresentar uma estimativa do valor final da compra;

d3) gerar uma tabela em CSV, com a informação da lista: um produto por linha, campos separados por ';'.

Questão 4: Gramáticas e Parsing (4v)

Considere a gramática independente de contexto, G, abaixo apresentada, que permite declarar uma ou mais variáveis definindo o seu tipo e permite executar instruções de dois tipos sobre essas variáveis. Note ainda que os símbolos terminais T e não-terminais NT estão definidos antes do conjunto de produções P, sendo L o seu axioma (ou símbolo inicial).

```
T  = { '{', '}', ';',  a, n, cmdB }
NT = { L, SE, SEL, C }
P = {
p1: L  -> SE ';'
p2:     | &
p3: SE -> a
p4:     | n
p5:     | '{' SEL '}'
p6: SEL-> SE C
p7: C  -> SEL
p8:     | &
}
```

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

a) Após estender a G dada, construa o respetivo **autómato LR(0)** e identifique todas as **situações de conflito** que eventualmente ocorram.

b) Se num dado momento do parsing Bottom-UP (BU) estiver num estado `q` e, ao ler o símbolo terminal da entrada `'}'`, a tabela de decisão `ACTION` indicar que deve fazer uma redução pela produção 5 (a ação determinada for `red#5`), diga quantos estados vai recuar no autómato de reconhecimentos (quantos símbolos tira da stack de parsing) e quais os símbolos por que pode transitar a seguir a reduzir.

c) Se num dado momento do `Parsing Top-Down LL(1)` a *stack de parsing* contiver os símbolos (topo à esquerda e `'$'` a representar o fim de ficheiro)

```
'}' | ';'  | $ |
```

diga o que significa esse estado, isto é, o que é que já foi reconhecido e qual pode ser o próximo símbolo do ficheiro de entrada para o parsing continuar sem erros.

d) Escreva as funções de um parser RD-puro (recursivo-descendente) para reconhecer os 2 símbolos não-terminais `SEL` e `C`.

Questão 5: Geração de Código (2v)

Recorde-se que nas aulas práticas foram definidas 5 variantes de `Instrução` para a linguagem imperativa *C-like*

```
inst: inst-atrib                { $$ = $1; }
    | inst-escrever             { $$ = $1; }
    | inst-ler                  { $$ = $1; }
    | inst-condicional          { $$ = $1; }
    | inst-ciclo                { $$ = $1; }
    ;
```

e imagine que agora se quer acrescentar uma nova instrução iterativa do tipo *repetir ... até*

```
inst:
.................
    | inst-repetir              { $$ = $1; }
    ;
inst-repetir: REPEAT '{' insts '}' UNTIL '(' cond ')'  { ASr }
    ;
```

Escreva então a ação semântica `ASr` para gerar o código Assembly da VM que implementa este novo ciclo.