

U.C. 21018

Compilação

27 de Setembro de 2016

-- INSTRUÇÕES --

- O estudante deverá responder à prova na folha de ponto e preencher o cabeçalho e todos os espaços reservados à sua identificação, com letra legível.
- No fim da prova, poderá ficar na posse do enunciado.
- Verifique no momento da entrega da(s) folha(s) de ponto se todas as páginas estão rubricadas pelo vigilante. Caso necessite de mais do que uma folha de ponto, deverá numerá-las no canto superior direito.
- Em hipótese alguma serão aceites folhas de ponto dobradas ou danificadas.
- Exclui-se, para efeitos de classificação, toda e qualquer resposta apresentada em folhas de rascunho.
- Os telemóveis deverão ser desligados durante toda a prova e os objectos pessoais deixados em local próprio da sala de exame.
- Utilize unicamente tinta azul ou preta.
- A prova é constituída por **3** páginas (esta página de rosto e duas com as questões), contém 5 questões, sem consulta, e termina com a palavra **FIM**. Verifique o seu exemplar e, caso encontre alguma anomalia, dirija-se ao professor vigilante nos primeiros 15 minutos da mesma, pois qualquer reclamação sobre defeito(s) de formatação e/ou de impressão que dificultem a leitura não será aceite depois deste período.

Duração: 150 minutos

1ª Questão (4 valores)

Considere a seguinte gramática:

```
L -> n | * [ E , L ]
E -> T % E | T
T -> F & T | F
F -> i | ( E )
```

Construa as tabelas de acções e saltos do analisador sintáctico ascendente LR, pelo método SLR.

2ª Questão (4 valores)

Apresente a especificação *flex* para um analisador léxico que identifique os tokens num documento XML:

- Tag inicial – começa por < e termina em >. No interior só pode ter letras e o hífen (-).
- Tag final – começa por </ e termina em >. No interior só pode ter letras e o hífen (-).
- Entidade – Uma das seguintes:
 - < que representa o símbolo <
 - > que representa o símbolo >
 - & que representa o símbolo &
 - ' que representa o símbolo ‘
 - " que representa o símbolo “

O programa deverá converter a notação XML para uma notação funcional, em que as tags correspondem a funtores e o texto entre as tags iniciais e finais é o argumento da função. Além disso, as entidades correspondentes a caracteres especiais devem ser convertidas no correspondente símbolo. Por exemplo, o documento XML

```
<expr> 3 &gt; 1 </expr>
```

deverá ser convertido em:

```
expr(3 > 1)
```

3ª Questão (4 valores)

Apresente o código intermédio, em notação TAC (*three address code*) correspondente ao seguinte excerto de código em linguagem C:

```
int a[25];
int i = 0;
int j = 0;

while (i < 5) {
    while (j < 5) {
        a[i*5+j] = i+j;
        j++;
    }
    i++;
}
```

4ª Questão (4 valores)

Apresente a especificação *bison* para um analisador sintático de expressões lógicas, que devolva o resultado da avaliação da expressão analisada. As constantes lógicas são identificadas pelo analisador léxico, sendo que o analisador sintático deverá considerar as seguintes operações:

- Conjunção: o resultado de $x \ \& \ y$ é **1** se x e y forem **1**. Caso contrário é **0**.
- Disjunção: o resultado de $x \ | \ y$ é **1** se x ou y forem **1**. Caso contrário é **0**.
- Negação: o resultado de $\sim \ x$ é **1** se x for **0** e é **0** se x for **1**.

Considere ainda que o operador $\&$ tem precedência sobre $|$ e que pode usar parêntesis para alterar a ordem de avaliação.

5ª Questão (4 valores)

Optimize o seguinte código gerado em TAC (*three address code*), explicitando o tipo de optimização que está a fazer:

```
a = 2 - 2
t1 = a / 2
t2 = t1 + 2
t3 = t1 * a
t4 = t2 + t3
t5 = t4 + b
t6 = t5 + t4
c = t6 * 2
```

FIM