

**U.C. 21018**

**Compilação**

**9 de setembro de 2014**

## **-- 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:

```
P -> Var Statement
Var -> tipo id ;
Statement -> id <- Expr ;
Expr -> Lua & Expr | Lua
Lua -> Terra # Lua | Terra
Terra -> ( Expr ) | id
```

Construa as tabelas de ações e saltos do analisador sintático ascendente LR, pelo método SLR.

## 2ª Questão (4 valores)

Apresente a especificação *flex* para um analisador léxico que identifique os seguintes tokens e imprima no ecrã o respectivo token ID e, entre parênteses, o respectivo valor.

- Identificador (token ID: VAR) – uma letra ou o carácter ‘\_’ seguida de letras, dígitos ou do mesmo carácter ‘\_’.
- Constante lógica (token ID: LOG) – **and**, **or**, **xor** ou **not**.
- Constante numérica (token ID: NUM) – uma sequência numérica, que pode ser inteira ou real, podendo no caso real ser com vírgula ou em notação científica.

## 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[10];
int i = 2;

a[0] = 0;
a[1] = 1;
while (i < 8){
    a[i] = i + a[i-1] + a[i-2];
    i++;
}
```

#### 4ª Questão (4 valores)

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

- Concatenação de strings: o resultado de  $s1 + s2$  é uma string com a sequência de caracteres de  $s1$  seguida da sequência de caracteres de  $s2$ . **Ex.** “abc”+”123” é “abc123”
- Repetição: o resultado de  $s ^ n$ , em que  $s$  é uma string e  $n$  é um número inteiro, é a concatenação de  $s$  com ela própria  $n$  vezes. Ex. “ab”<sup>3</sup> é “ababab”

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 otimização 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 = t5 * 2
```

**FIM**