

# Primeira Prova de MAB 471 2013.2 — Compiladores I

Fabio Mascarenhas

16 de Outubro de 2013

A prova é individual e sem consulta. Responda as questões na folha de respostas, a lápis ou a caneta. Se tiver qualquer dúvida consulte o professor.

Nome: \_\_\_\_\_

DRE: \_\_\_\_\_

|          |   |   |   |       |
|----------|---|---|---|-------|
| Questão: | 1 | 2 | 3 | Total |
| Pontos:  | 4 | 2 | 4 | 10    |
| Nota:    |   |   |   |       |

1. Algumas linguagens de programação possuem numerais complexos. Um numeral complexo pode ser um numeral imaginário, dado por um numeral decimal seguido por  $i$  ou  $j$  (Ex:  $123i$ ,  $5.5j$ ), ou uma parte real dada por um numeral decimal seguida por um operador  $+$  ou  $-$  e um numeral imaginário, podendo haver espaços em volta do operador (Ex:  $1+2i$ ,  $3.5 - 4.0j$ ).

- (a) (2 pontos) Escreva a(s) regra(s) léxica(s) para numerais complexos, usando expressões regulares.  
(b) (2 pontos) Escreva um autômato finito determinístico para numerais complexos.

2. A gramática a seguir dá uma outra formulação para a parte da gramática de TINY que descreve comandos e sequências de comandos:

```
CMD -> CMD ; CMD
      | if EXP then CMD end
      | if EXP then CMD else CMD end
      | repeat CMD until EXP
      | id := EXP
      | read id
      | write EXP
```

- (a) (1 ponto) Mostre que a gramática é ambígua (pode tratar  $EXP$  como um terminal).  
(b) (1 ponto) Reescreva essa gramática para eliminar a ambiguidade, fazendo a sequência  $(;)$  ser associativa à direita.

3. A gramática a seguir descreve uma linguagem de S-expressões (estilo Lisp):

```
SEXP -> ATOMO | LISTA
ATOMO -> num | id
LISTA -> '(' SEXPS ')
SEXPS -> SEXP SEXPS | SEXP
```

- (a) (1 ponto) Escreva a derivação mais à esquerda para o programa `( id ( id ( num ) ) ( id ) )`.
- (b) (1 ponto) Escreva o pseudocódigo para o analisador recursivo sem retrocesso dos não-terminais `SEXP`, `ATOMO` e `LISTA`. Lembre-se de construir a árvore corretamente.
- (c) (2 pontos) Por que essa gramática não é LL(1)? Reescreva a gramática mudando apenas o necessário para que ela se torne LL(1), e escreva o pseudocódigo para o analisador recursivo sem retrocesso da parte que foi reescrita.

BOA SORTE!