BCC328 — Construção de Compiladores 1
Prova 3
Matrícula:
Nome:

DECOM UFOP 12/02/2014
Prof. José Romildo Malaquias

Questões

Questão 1. [2 PONTOS]
Considere a seguinte gramática:

$S \rightarrow A$
$A \rightarrow B$
$A \rightarrow C$
$B \rightarrow$ `(` C `)`
$C \rightarrow B$ `+` C
$C \rightarrow D$
$D \rightarrow$ `0`
$D \rightarrow$ `1`

Quais das seguintes regras de produção tornam esta gramática recursiva à esquerda? (Escolha todas que se aplicam.)

1. $D \rightarrow B$
2. $B \rightarrow C$
3. $D \rightarrow A$
4. $A \rightarrow D$
5. $C \rightarrow C$ `+` B
6. $C \rightarrow$ `1` C

Questão 2. [2 PONTOS]
Considere a seguinte gramática:

$S \rightarrow A$ `a`
$S \rightarrow$ `c`
$A \rightarrow S$ `b`

Quais das seguintes gramáticas remove corretamente a recursividade à esquerda desta gramática? (Escolha todas que se aplicam.)

1. $S \rightarrow A$ `a`
   $S \rightarrow$ `c`
   $A \rightarrow$ `c` `b` A'
   A' $\rightarrow$ `a` `b` A'
   A' $\rightarrow$
2. $S \rightarrow A$ `a`
   $S \rightarrow$ `c`
   $A \rightarrow$ `c` `b`
   $A \rightarrow A$ `a` `b`
3. $S \rightarrow$ `c` A `a`
   $S \rightarrow$ `c`
   $A \rightarrow A$' A
   $A \rightarrow$
   A' $\rightarrow$ `a` `b`
4. $S \rightarrow$ `c` S'
   S' $\rightarrow A$ S'
   S' $\rightarrow$
   $A \rightarrow$ `b` `a`

Questão 3. [4 PONTOS]
Considere a seguinte gramática para expressões pós-fixas:

$E \rightarrow E$ E `+`
$E \rightarrow E$ E `*`
$E \rightarrow$ `num`

Construa a tabela de análise LL(1) adequada a um analisador sintático preditivo para reconhecer cadeias da linguagem gerada por esta gramática.

Se necessário:

- primeiramente elimine a recursividade à esquerda da gramática dada, e
- faça a fatoração à esquerda da gramática resultante, e em seguida
- acrescente uma nova regra de produção com um novo símbolo inicial onde o lado direito da regra consiste no antigo símbolo inicial seguido do terminal que indica fim da entrada.

Questão 4. [2 PONTOS]
Descreva a análise semântica da expressão de repetição `loop` que foi adcionada à linguagem Panda como

uma extensão pelos alunos de *Construção de Compiladores*.

A expressão loop é da forma

```
loop var1 := init1 =: step1;
     ⋮
     varn := initn =: stepn
when test => result;
     body
```

onde var_1, ..., var_n são identificadores e representam variáveis, $init_1$, ..., $init_n$, $step_1$, ..., $step_n$, *test*, *result* e *body* são expressões, e `loop` e `when` são palavras reservadas.

Quando avaliada, a expressão loop

1. avalia as expressões $init_1$, ..., $init_n$ nesta ordem, e estende o ambiente com as novas variáveis textitvar$_1$, ..., var_n, inicializadas com os valores obtidos. O escopo destas variáveis se estende sobre $step_1$, ..., $step_n$, *test*, *result* e *body*;
2. em seguida a expressão *test* é avaliada
   (a) se o seu valor for verdadeiro, *result* é avaliada e o seu valor é o valor da expressão *loop*;
   (b) se o valor de *test* for falso, a expressão *body* é avaliada, e em seguida as expressões $step_1$, ..., $step_n$ são avaliadas nesta ordem, e os seus valores são atribuídos às variáveis var_1, ..., var_n, respectivamente; a avaliação continua no passo 2.

Exemplo:

```
loop x := 10-1 =: x+1;
     y := 100  =: 2*y + x
when y <= 0 => 2*(x+1);
   print_int(x)
```