Compiladores: Prova 1

Nome:

Matrícula: Data:

Observações:

(a) A prova é individual e sem consulta, sendo vedado o uso de calculadoras e de telefones celulares.

(b) A interpretação dos comandos das questões faz parte da avaliação.

(c) A nota da prova será igual a 0 (zero) caso o estudante consulte algum material durante a prova, ou receba ou ofereça qualquer ajuda a outro estudante durante a prova.

(d) As respostas podem ser redigidas à lapis. Respostas sem justificativas não serão consideradas e a respectiva questão terá nota 0 (zero).

(e) O grampo da prova não deve ser removido. Caso o grampo seja removido, a nota da prova será igual a 0 (zero).

Parte A. (15 pontos) Resposta às questão que se seguem.

1. As fases da compilação podem ser agrupadas em partes. Quantas e quais são estas partes? Para cada parte, indique as fases que a compõe e destaque a principal característica comum a todas estas fases.

Solução:
A compilação pode ser dividida em duas partes: análise e síntese. A parte de análise é composta de 3 fases: análise léxica, análise sintática e análise semântica; a parte de síntese é composta de 3 fases: geração de código intermediário, otimização e geração de código.

As fases da análise dizem respeito à extração, organização e classificação das informações contidas no programa fonte; já as fases da síntese tratam da produção de novos códigos que culminem no programa alvo.

2. Sejam $\odot$ e $\oslash$ dois operadores binárias cujos operandos e resultados pertençam ao conjunto $\Sigma = \{ a, b, c \}$. Sabendo que $\odot$ é associativa à direita, $\oslash$ é associativa à esquerda e que $\oslash$ tem maior precedência entre ambos, defina uma gramática livre de contexto, sem ambiguidades, cujas sentenças sejam expressões envolvendo ambos operadores. Não é necessário incluir suporte para parêntesis.

Solução:

$$\begin{aligned} expr &\rightarrow expr \oslash fator \mid fator \\ fator &\rightarrow termo \odot fator \mid termo \\ termo &\rightarrow a \mid b \mid c \end{aligned}$$

3. Observe as informações abaixo.

$$\begin{aligned} S &\rightarrow aA \{ imprimir(a) \} \\ S &\rightarrow \{ imprimir(b) \} bB \\ A &\rightarrow bS \\ B &\rightarrow c \end{aligned}$$

Estas informações definem uma tradução dirigida pela sintaxe ou um esquema de tradução? Justifique sua resposta.

Solução:
As informações (produções) dadas definem um esquema de tradução, uma vez que em algumas destas produções são inseridas, entre parêntesis e em locais específicos, ações semânticas.

Parte B. (5 pontos) Resolva a questão a seguir.

4. Considere a gramática livre de contexto G dada abaixo:

$$X \rightarrow aX \mid bYX$$
$$Y \rightarrow aX \mid bXY \mid \epsilon$$

Implemente, em C, C++ ou Python, as rotinas associadas aos não-terminais de G em uma implementação de um analisador gramatical recursivo descendente para G. Assuma que as rotinas `reconhecer`($token$) e `erro`( ) já estejam devidamente implementadas e disponíveis para uso, assim como a variável global $lookahead$. Na implementação, considere que a e b sejam representados pelos caracteres ASCII `'a'` e `'b'`, respectivamente.

Importante: Escreva o código com letra legível, de forma organizada e clara, numerando as linhas. O código não deve exceder 50 linhas.

```
1  void X()
2  {
3      switch (lookahead) {
4      case 'a':
5          reconhecer('a');
6          X();
7          break;
8
9      case 'b':
10         reconhecer('b');
11         Y();
12         X();
13         break;
14
15     default:
16         erro();
17     }
18 }
19
20 void Y()
21 {
22     switch (lookahead) {
23     case 'a':
24         reconhecer('a');
25         X();
26         break;
27
28     case 'b':
29         reconhecer('b');
30         X();
31         Y();
32     }
33 }
```

```
1  def X():
2      if lookahead == 'a':
3          reconhecer('a');
4          X()
5      elif lookahead == 'b':
6          reconhecer('b');
7          Y()
8          X()
9      else:
10         erro()
11
12
13 def Y():
14     if lookahead == 'a':
```

```
15          reconhecer('a');
16          X()
17      elif lookahead == 'b':
18          reconhecer('b');
19          X()
20          Y()
```