Pontifícia Universidade Católica de Minas Gerais
ICEI - Departamento de Ciência da Computação
Disciplina: Compiladores
Prof. Alexei Machado

Lista de Exercícios No. 1

1. Dada a gramática abaixo, dê as derivações mais a esquerda e árvores de derivação dos strings *aabbba* e *abbbaa*.

$S \rightarrow ASB \mid \lambda$
$A \rightarrow aAb \mid \lambda$
$B \rightarrow bBa \mid ba$

2. Dada a gramática abaixo, dê as derivações mais a esquerda e árvores de derivação dos strings *aacacaacb* e *aacb*.

$S \rightarrow aA \mid c$
$A \rightarrow acA \mid b \mid S$

3. Dada a gramática abaixo, escreva seu grafo mais a esquerda para verificar se os strings *aabcc* e *abbacc* pertencem à linguagem.

$S \rightarrow aS \mid A$
$A \rightarrow bA \mid B$
$B \rightarrow cB \mid c$

4. Para cada gramática abaixo, gere outra gramática equivalente sem recursão direta à esquerda e fatorada:

   a) $S \rightarrow abA \mid acS$
      $A \rightarrow Ac \mid b$

   b) $S \rightarrow C \mid A$
      $A \rightarrow AaB \mid Aac \mid B \mid a$
      $B \rightarrow Bb \mid Cb$
      $C \rightarrow cC \mid c$

   c) $S \rightarrow AS \mid A$
      $A \rightarrow id := E$
      $E \rightarrow E+num \mid E + id \mid id \mid num$

5. Para cada a gramática abaixo, implemente um parser descendente preditivo para a linguagem, se possível:

   a) $S \rightarrow aAb \mid b$
      $A \rightarrow aA \mid B$
      $B \rightarrow bB \mid \lambda$

   b) $S \rightarrow SaSb \mid SbSa \mid \lambda$

   c) $S \rightarrow IF$
      $I \rightarrow 1I \mid 0I \mid \lambda$
      $F \rightarrow .D \mid \lambda$
      $D \rightarrow 1D \mid 0D \mid E$
      $E \rightarrow 1 \mid 0$

d) $S \rightarrow bcA \mid bcd \mid Ba \mid d$
$A \rightarrow Ab \mid Ac \mid d$
$B \rightarrow bc$

6. Considere o comando de concatenação de arquivos CONCAT, cuja sintaxe é:

CONCAT arq1 + arq2 + . . . + arqn > arqd

onde arq1 ... arqn e arqd são representados pela expressão regular (id: ∪ λ)(λ ∪ \(id\)*)id(.id ∪ λ), sendo id um token da linguagem com padrão de formação l(l∪d)*. A lista de arquivos de origem precisa conter pelo menos um arquivo.

a) Indique qual é o alfabeto da linguagem e o padrão de formação dos lexemas
b) Escreva o analisador léxico para o problema.
c) Escreva uma gramática livre de contexto adequada ao parsing descendente predizível para o comando.
d) Implemente o parser correspondente.
e) Escreva uma gramática com expressões regulares adequada ao parsing descendente predizível para o comando.
f) Implemente o parser correspondente.

7. Considere um subconjunto da linguagem C++ que trata da declaração de classes. A declaração de uma classe tem a seguinte forma:

class *nome base { membros };*

onde *class* é uma palavra reservada, *nome* é o identificador da classe, *base* é uma lista opcional de classes para definição de herança e *membros* é uma seqüência de zero ou mais declarações de atributos e métodos da classe. Existindo herança, *base* tem a seguinte estrutura:

: *acesso classe_1****,*** *acesso classe_2****, ... ,*** *acesso classe_n*

onde *acesso* pode assumir os valores **private**, **protected** ou **public**, que são palavras reservadas da linguagem, e *classe_1... classe_n* são identificadores de classes. As formas de acesso também são opcionais (o default é private).

Membros de classe podem ser declarações de atributos, cuja sintaxe é idêntica à utilizada para declaração de variáveis (considere apenas os tipos **int** e **float** sem inicialização), ou declarações de métodos da forma:

retorno nome(lista_parâmetros);

onde *retorno* pode assumir as palavras reservadas **void, int** ou **float,** *nome* é um identificador e *lista_parâmetros* é uma seqüência de zero ou mais parâmetros separados por vírgulas, cujo tipo deve ser indicado mas seu identificador é opcional. Considere apenas os tipos **int** e **float.** Dois métodos têm sintaxe especial: o construtor da classe não possui retorno e o destrutor não possui nem retorno nem parâmetros, além de ter o símbolo **~** antes do identificador, o qual deve ser igual ao nome da classe.
As declarações de membros podem ser precedidas de indicação de acesso:

acesso:

onde acesso assume os valores **private**, **protected** ou **public.** Identificadores podem ser formados de letras, dígitos e sublinhado, devendo começar com letra ou sublinhado. O sublinhado sozinho não será considerado válido. Comentários são delimitados por /* */ ou começam com // e terminam com a quebra de linha.

Exemplo:

```
Class X : A, public B
{
   int a;
   float b,c;
   float area(float lado, int);
   public:
      ~X();                                  // destrutor
      X(float, int);                         /* construtor */
};
```

a) Defina o alfabeto para o subconjunto da linguagem e forneça o padrão de formação dos lexemas.
b) Escreva um AFD que represente o analisador léxico para a linguagem.
c) Escreva uma gramática LL(1) para a linguagem.
d) Implemente o parser correspondente

8. O UNIX é um dos sistemas operacionais mais utilizados no mundo. Uma de suas interfaces, o C shell, constitui uma poderosa linguagem de programação. A linguagem na qual se baseia esta questão foi inspirada no C shell. Declarações e comandos podem aparecer em qualquer parte do programa, cada um terminado por uma quebra de linha. Se um comando precisar ser continuado em outra linha, deve-se utilizar a barra invertida para indicar que o comando não acabou. Comentários são iniciados por # e terminam com a quebra de linha. Identificadores são constituídos por letras, dígitos e sublinhado, não podem começar com dígitos e devem ter pelo menos uma letra ou um dígito. Números reais usam o ponto decimal, podem começar com o ponto mas precisam ter parte fracionária. A seguir é descrita a sintaxe para declarações de inteiros e reais, comandos de atribuição e estruturas de teste:

- Declarações: São da forma ***set id=num*** onde *id* é um identificador de variável e *num* uma constante inteira ou real.
- Atribuições: São da forma ***@id=exp*** onde *exp* é uma expressão.
- Expressões: São formadas com operadores **+**, **-**, ***** e **/** , respeitando-se a precedência, números e conteúdos de identificadores. Para se ter acesso ao conteúdo de um identificador, deve-se anexar o símbolo **$** antes dele. Parênteses são permitidos.
- Testes: São da forma ***if (exp R exp) comando*** onde *R* é um relacional (**<**, **>**, **==**) e *comando* um comando único da linguagem. Se o teste possuir a parte do *senão*, deverá vir na próxima linha com a sintaxe ***else comandos endif*** onde *comandos é* uma seqüência de um ou mais comandos, cada um em uma linha. Se a parte do *então* contiver mais de um comando, deverá seguir a seguinte sintaxe: ***if (exp R exp) then*** . Os comandos vêm a partir da próxima linha e a seqüência termina com a palavra reservada ***endif*** ou o ***else*** se houver a parte do *senão*.

Exemplo:

```
############################ Este e' um script
set i=0                     # declara e inicializa
@i = $i+3*2                 # soma 6 ao conteudo de i
set \
  j=5.0                     # declara j em duas linhas
if ($i>0) @i=2              # teste se entao unico
if ($i==2) @j=1.0
else @j=2.0
endif                       # teste se entao senao unico
if ($j>1.0) then
  set k=0
```

```
  @i=2*(1+$j)
else @i=1
endif                        # entao multiplo
if ($j>0) if ($i>1) then  # ifs aninhados
    @i=1
    @j=1
  endif
```

a) Defina o alfabeto para o subconjunto da linguagem e forneça o padrão de formação dos lexemas. Maiúsculas e minúsculas são diferenciadas.
b) Escreva um AFD que represente o analisador léxico para a linguagem.
c) Escreva uma gramática LL(1) para a linguagem.
d) Implemente o parser completo, baseado na gramática do item anterior.