Primeira Prova de Construção de Compiladores
Primeiro Semestre de 2006, DC-UFSCar
Prof. José de Oliveira Guimarães
Turma B (Quinta feira).

1.(6.0) Abaixo é mostrado parte da gramática de uma linguagem.

```
Program ::=  Command { Command }
Command ::= IfCommand | … | ForEachCommand
ForEachCommand ::= “foreach” Id “in”  Id “do” Command
```

A partir destas regras, faça:
a) (2.5) as classes Command e ForEachCommand da ASA. Estas classes devem ter um método para gerar código em C. Não é necessário fazer o construtor, mas faça todos os outros métodos. Não se esqueça das heranças. Você pode assumir que existe apenas um comando foreach em todo o programa e que nomes de variáveis nesta linguagem nunca começam com sublinhado (_). Estas observações o ajudaram na hora de gerar código em C.
b) (3.5) o método forEachCommand do analisador sintático. Naturalmente, construa a ASA durante esta análise e faça a análise semântica. O primeiro Id do comando foreach deve ser uma variável de um tipo básico. O segundo Id deve ser uma variável cujo tipo é um vetor e os elementos deste vetor devem ter tipo igual ao tipo da primeira variável. Por exemplo, em

```
foreach x in v do  write(x);
```

se v tiver sido declarado como “var v : array[1..10] of integer;”, então o tipo de x deve ser integer. Admita que a ASA possua uma classe ArrayType subclasse de Type com os seguintes métodos: int getLimInferior(), int getLimSuperior(), Type getElemType(). Estes métodos retornariam, para o objeto da ASA que representa o tipo de v, 1, 10 e um ponteiro para o objeto apontado por Type.integerType. O comando foreach acima imprime todos os elementos do vetor v.

Você possivelmente usará a classe Type. Esta possui subclasses IntegerType, CharType, BooleanType e ArrayType. Type possui variáveis estáticas integerType, charType e booleanType. Cada uma delas aponta para o único objeto do programa de cada subclasse de Type que é tipo básico. A tabela de símbolos é referenciada pela variável st e possui métodos Object put(Object name, Object value) e Object get(Object name). O primeiro insere um símbolo e o segundo faz a busca por símbolo.

2. (3.5)
a) (1.5) Faça uma tabela de símbolos a ser utilizada em um compilador de Java. **Assuma** que existem três níveis léxicos: um para classes, um para métodos e variáveis de instância e outro para variáveis locais e parâmetros. Faça a classe que é a tabela de símbolos com as interfaces dos métodos e a declaração das

variáveis de instância. Naturalmente, coloque nomes dos métodos e das variáveis de instância auto-explicativos.
b) (2.0) Explique, utilizando um exemplo de código em Java, quando é feita uma busca e quando é feita uma eliminação de identificador da tabela de símbolos. Isto é, explique **porquê** é feita a busca em certo ponto (mostre o local) e **porquê** é feita a eliminação (e mostre o local).

3. (2.5) Complemente o seguinte analisador léxico para que ele :
a) elimine comentários que começam com // e terminam no final da linha. Não se esqueça de que o último caráter do vetor `in` é '\0';
b) considere os caracteres '\n' e '\r' como espaços em branco;
c) reconheça os símbolos <= e <. Para o primeiro, token deve receber Symbol.LE e, para o segundo, Symbol.LT.

```
void nextToken() {
  while ( in[p] == ' ' )
     p++;
  if ( in[p] == '\0' ) { token = Symbol.EOF; return ; }
  if(Character.isDigit(in[p]))
         reconheceDigito();
  else
    switch ( in[p] ) {
      case '+' : token = Symbol.PLUS; p++; break;
      …
    }
}
```

Copie o código acima na sua resposta.