Primeira Prova de Construção de Compiladores
Primeiro Semestre de 2003, DC-UFSCar
Prof. José de Oliveira Guimarães
Turma B (Quinta feira).

1.(5.5) Dada a gramática

```
Program ::=  Command { Command }
Command ::= SimpleCommand | RepeatCommand
SimpleCommand ::= 'a' Character | BackCommand | 'f' Number | 'd'
BackCommand ::= 'b' Number
Character ::= qualquer caráter
Number ::= qualquer dígito
RepeatCommand ::= 'r' Number SimpleCommand
```

Faça:
a) os métodos program, command, backCommand e repeatCommand do analisador sintático. Naturalmente, construa a ASA durante esta análise. No início de backCommand e repeatCommand, assuma que o token corrente seja 'b' e 'r', respectivamente.
b) as classes Program, Command, BackCommand e RepeatCommand. ***Não*** é preciso fazer ***nenhum*** método. Não se esqueça das heranças.

2. (1.5) Cite cinco (5) possíveis erros léxicos em um programa em Java. Cite exemplos.

3. (3.0) Esta questão se refere à linguagem do compilador 10, onde há dois escopos diferentes. A tabela de símbolos possui a seguinte interface:

```
public class SymbolTable {
    ...           // coloca (key, value) na tabela global
    public Object putInGlobal( String key, Object value )
    public Object putInLocal( String key, Object value )
    public Object getInLocal( Object key )
    public Object getInGlobal( Object key )
       // get busca na tabela local e então na global
    public Object get( String key )
    public void removeLocalIdent()
}
```

Nos compiladores estudados nesta disciplina, utilizamos objetos para representar os tipos. Poderia ser diferente: as constantes Symbol.INTEGER, Symbol.CHAR e Symbol.BOOLEAN poderiam representar os tipos. Por exemplo, a classe `Variable` teria uma variável de instância "`int type`" e `getType` simplesmente retornaria `type`. Na classe `Compiler`, a TS é referenciada pela variável `st`. Assumindo isto, faça as conferências semânticas e colocação de variáveis na ASA dos seguintes itens:
a)

```
    private Variable varDec() {
        Variable v;
        if ( token != Symbol.IDENT )
```

```
          error("Identifier expected");
        String name = stringValue; // name of the identifier
        nextToken();
        // faça a análise semântica aqui.
        // Coloque a variável na TS.
        ...
        return v;
    }
```

b)

```
    // assuma que não existe tipo undefinedType
    private Expr addExpr() {
      int op;   Expr left, right;
      left = multExpr();
      while ( (op = lexer.token) == Symbol.PLUS ||
                op == Symbol.MINUS ) {
        lexer.nextToken();
        right = multExpr();
        ... // coloque aqui a AS
        left = new CompositeExpr( left, op, right );
      }
      return left;
    }
```

4. (1.0) Faça o código em Java que cria os objetos da ASA do exemplo abaixo. As instruções deverão ser inseridas em um vetor statList (chamando o método addElement). Assuma que `read` pode ler apenas uma variável por vez.

```
read(n);
while n < 100 do
   n = n + 1;
```

6. (1.0) O programa abaixo possui uma falha que seria apontada por alguns compiladores embora ele compile em outros. O programa imprime 2 e um número diferente de -6. Por quê ? Explique todo o seu raciocínio --- não é suficiente apenas apontar algo estranho ou errado.

```
#include <iostream.h>

class X  {
   public:
      unsigned x;
      X( unsigned k ) { x = k; }
      X() { x = 0; }
      operator unsigned() { return x; }
}

twice( int n )   { return 2*n; }

int main() { cout << twice(1) << ',' << twice(-3); }
```