Primeira Prova de Construção de Compiladores.
Turma B. Quinta Feira.
Prof. José de Oliveira Guimarães.

1 (5.0) O comando for de C tem o seguinte formato:

```
for (expression; expression; expression ) statement
```

Um analisador sintático incompleto para este comando é dado abaixo. Copie este analisador para a folha de respostas acrescentando o restante da análise sintática e o código para a criação do objeto da ASA para o for. Faça também a classe da ASA ForStatement com o método genC para gerar código.

```
ForStatement forStatement() {
   nextToken();
   if ( lexer.token != Symbol.LEFTPAR ) error();
   nextToken();
   Expr e1 = expr();
   if ( lexer.token != Symbol.SEMICOLON )
      error();    //   semicolon é ;
   nextToken();
   Expr e2 = expr();
   if ( lexer.token != Symbol.SEMICOLON )
      error();    //   semicolon é ;
   nextToken();

}
```

2. (3.0) A classe Type possui subclasses IntegerType, CharType e BooleanType. Type possui variáveis estáticas integerType, charType e booleanType. Cada uma delas aponta para o único objeto do programa de cada subclasse de Type. Baseado nisto, faça as conferências semânticas da atribuição e comando if (completo o código - copie o código abaixo para a folha de respostas. A tabela de símbolos é referenciada pela variável st e possui métodos put(Object name, Object value) e Object get(Object name). O primeiro insere um símbolo e o segundo faz a busca por símbolo.

```
    private AssignmentStatement assignmentStatement() {
        // neste ponto lexer.token == Symbol.IDENT
        String name = (String ) lexer.getValue();

        if ( lexer.getToken().tk != Symbol.ASSIGN )
          error.show("= expected");
        else
          lexer.nextToken();
        Expr right = orExpr();
        return new AssignmentStatement( v, right );
    }

    private IfStatement ifStatement() {

        lexer.nextToken();
        Expr e = orExpr();
        if ( lexer.getToken().tk != Symbol.THEN )
```

```
          error.show("then expected");
        else
          lexer.nextToken();
        ... // nao interessa.
    }
```

3.(2.0) Copie o analisador léxico abaixo na folha de respostas e o modifique para que ele reconheça comentários como os de Pascal, delimitados por { e }. Não se esqueça de sinalizar o erro “comentário aberto e não fechado”. Lembre-se de que após reconhecer o comentário, nextToken() deve ser recursivamente chamado.

```
public void nextToken() {
   char ch;

   while (  (ch = input[tokenPos]) == ' ' || ch == '\r' ||
                ch == '\t' || ch == '\n')
          tokenPos++;
   if ( ch == '\0')
      token = new Symbol(Symbol.EOF);
   else
        // coloque o seu código aqui.
      if ( Character.isLetter( ch ) ) {
                // get an identifier or keyword
         ...
      }
      else {
         ...
      }
}
```

4. (1.0) Faça a representação gráfica da ASA para o seguinte trecho de programa:

```
   for i = 1 to n do
      write(i);
```