1.

```
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();
   Expr e3 = expr();
   if ( lexer.token != Symbol.RIGHTPAR )
      error();    //   semicolon é ;
   nextToken();
   Statement s = statement();
   return new ForStatement(e1, e2, e3, s);
}

public class ForStatement extends Statement {
   public ForStatement( Expr startExpr,
                        Expr testExpr,
                        Expr endExpr,
                        Statement statement ) {
      this.startExpr = startExpr;
      this.testExpr = testExpr;
      this.endExpr = endExpr;
      this.statement = statement;
   }

   public void genC( PrintWriter out ) {
      out.print("for ");
      startExpr.genC(out);
      out.print("; ");
      testExpr.genC(out);
      out.print("; ");
      endExpr.genC(out);
      out.println(" )");
      statement.genC(out);
   }

   private Expr startExpr, testExpr, endExpr;
   private Statement statement;
}
```

2.

```
    private AssignmentStatement assignmentStatement() {

        String name = (String ) lexer.getValue();
        Variable v = (Variable ) symbolTable.get(name);
        if ( v == null )
            error.signal("Variable " + name + " was not
declared");
        lexer.nextToken();
        if ( lexer.getToken().tk != Symbol.ASSIGN )
           error.signal("= expected");
        else
           lexer.nextToken();
        Expr right = orExpr();
        if ( v.getType() != right.getType() )
          error.signal("Type error in assignment");

        return new AssignmentStatement( v, right );
    }

    private IfStatement ifStatement() {

        lexer.nextToken();
        Expr e = orExpr();
        if ( e.getType() != Type.booleanType )
          error.signal("Boolean type expected in if
expression");

        if ( lexer.getToken().tk != Symbol.THEN )
           error.signal("then expected");
        else
           lexer.nextToken();
        ...
    }
```

3.

```
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 ( ch == '{' ) {
        tokenPos++;
        while ( input[tokenPos] != '}' && “
                input[tokenPos] != '\0' )
           tokenPos++;
        if ( input[tokenPos] == '}' )
           tokenPos++;
        else
           error.signal("Comentário aberto e não fechado");
        nextToken();
      }
      if ( Character.isLetter( ch ) ) {
                // get an identifier or keyword
        ...
      }
      else {
        ...
      }
   }
}
```

4.