Open book exam. Duration: 1h30m First Midterm Exam ("Mini-Teste")

Group 1. *First*, *Follow* Sets and *Parsers* LL(k) (5 pts)

Consider the CFG[1] below.

```
S → AT
A → aAa | bAb | #T
T → aT | bT | ε
```

1.a) [1pt] Give the *First* and *Follow* sets for each grammar variable;

1.b) [2pts] Show the table for the parser LL(1);

1.c) [2pts] Indicate the possible problems this grammar may have and that need to be solved in order it can be implemented as a top-down recursive parser LL(1).

Group 2. Lexical and Syntactic Analysis (6 pts)

We intend to design and implement a programming language. The presented CFG G1 below represents the current version of the grammar for the language.

Some of the Tokens for G1:
IDENTIFIER = [a-zA-Z][0-9a-zA-Z]*
WHILE = while
ENDWHILE = endwhile
DO = do
IF = if
ELSE = else
THEN=then
ENDIF = endif

2.a) [0.5pt] Considering that OP represents the arithmetic operations, -, +, *, /, and CMP the comparison operations, !=, ==, >, <, >=, <=, show the definitions of these tokens as regular expressions.

2.b) [1pt] Show the DFA for the tokens WHILE, ENDWHILE, IF, ELSE, and ENDIF.

2.c) [1pt] Show the concrete syntax trees for the code example below and considering the depicted CFG.

2.d) [0.5pt] Show a possible abstract syntax tree (AST[2]) for the concrete syntax tree of the previous question.

Grammar G1:
1. S → (Stmt)*
2. Stmt → While | Assign | If
3. While → WHILE Cond DO (Stmt)* ENDWHILE
4. Assign → IDENTIFIER = Operand (OP Operand)? ;
5. Operand → IDENTIFIER | CONST
6. If → IF Cond THEN Stmt (ELSE Stmt)? ENDIF
7. Cond → Operand CMP Operand

2.e) [2pt] Show the necessary functions, considering the grammar rules of lines 2 to 3, and the respective pseudo-code to implement them, as a top-down recursive LL(1) parser,. Assume the existence of the lexical analyzer which outputs the sequence of tokens, the existence the global variable *token*, and the function *next()* which returns the next token in the sequence of tokens (in the beginning the variable *token* identifies the first token in the sequence).

Code example:
```
A=3;
while A >= 0 do
  A = A – 1;
endwhile
```

2.f) [1pt] Suppose we want to modify the grammar in order to make possible to input values from the keyboard and print values of variable to the screen. Present the grammar modifications you suggest.

Group 3. Semantic Analysis (5 pts)

Consider the segment of Java code of the example below.

[1] *Context-Free Grammar*
[2] *Abstract Syntax Tree*

Example

```
1.  public int f1(int B[]) {
2.      int[] A = {1, 2, 3, 4};
3.      int S = 0;
4.      int i;
5.      for (i = 0; i < A.length; i++) {
6.         S += A[i]*B[i];
7.      }
8.      return S;
9.  }
```

3.a) [2pts] Assuming that the variables used in the code can only be parameters or local variables, show a possible symbol table for this example indicating the information to be included in the descriptors.

3.b) [2pts] Show a high-level intermediate representation for the example, using as reference the high-level intermediate representation based on expression trees and presented in the classes of the course.

3.c) [1pt] In some programming languages, such as Java, the indexing of arrays neither can have negative values nor values greater than N-1 (with N the size of the specific array dimension being indexed). Discuss the inclusion of this kind of verification in the semantic analysis of a compiler, mentioning the possible problems and solutions.

Group 4. General (4 pts)

Comment the following sentences, indicating if they are true or false, and justifying your answers (if helpful include illustrative examples to help on justifying your answers):

4.a) [2pts] "The existence of ambiguity in a given CFG always implies conflicts in its LL(1) parser table.".

4.b) [2pts] "Compilers split the lexical, syntactic and semantic analysis in separated stages because it is impossible to implement them in a single stage.".

(End.)