**PARSER FOR PL0**

**CMS 395-1**

**SYNTAX ANALYZER**

**ROLLINS COLLEGE**

**Ramon A. Mata-Toledo Ph.D.**

The purpose of the syntax analyzer or parser is to check that the syntax of the program is correct. That is, that it follows the rules of the language. You must assume that the Lexical Analyzer did his job correctly. You can use the same program that we used to test your syntax analyzer. That program is correct according to the rules of the grammar described by the railroad diagrams. For testing your programs, I will alter that program by introducing statements that do not follow the diagram. I will not include any of the errors that I did for the lexical analyzer. To test this parser, for example, I will include two consecutive identifiers without separating them with a comma. Also having incomplete statement such as identifier =; and errors like that. So, use your imagination to test your own programs.

Input: There are two different ways to provide the input to the parser. The first one is to read the next token from the file that was generated by the syntax analyzer. You may have to remove the \*s that were used for making the lines easy to read. So, what you have as input is a bunch of integer numbers. Each integer numbers indicates a type of token according to the specifications of the language and the convention of numbers that we assign to them.

The second way to provide input to the parser is to have your previous program, the lexical analyzer, become a “slave” to the parser providing the next token to analyze. That is, the parser requests the next token from the lexical analyzer.

You are free to use the method that you find easier to work with, however, compilers use, in general, the master-slave mode. You must keep in mind that if the parser is expecting an identifier, it must “see” an “identifier” otherwise it an error. Just as we indicated when we talked about the recursive descent parser in class, the parser does not check at all that the identifier is in the symbol table or if it masks another identifier. Although I asked you to do that for the lexical analyzer is not the job of the parser to do so although it may be incorporated in the lexical analyzer as I asked you to do.

**Note No. 1**: If you read Hartmann’s method, you will see that he divides his compiler in 11 phases, each phase is dedicated to a different task. So, in his case, he reads for each program 10 input different files looking of syntax and semantic errors which we do not do here. We would need a second compiler class to do this and generate code.

**Hint**: The best way to write your parser is to have a bunch of constants, each for each type of token. For example, a constant\_EOF = 35 (if this is the number, we assigned to EOF), Another constant could be comma\_constant = 5 (if this is the number assigned to the comma). Every time that the parser encounters an error it should write, for example, “Error: missing identifier. I was expecting an identifier but found a semicolon instead.” For a recursive descent parser this is a very easy task because you have the current token and what you were expecting to see. If your program finds an error, it should try to recover by finding a token that belongs to the FOLLOW set and skipping the rest of the tokens until it finds one that is in the FOLLOW set, then try to continue parsing. Your parser may or not recover well. I just want you to see your parser attempting to recover. You can use Hartmann’s three basic schemes to facilitate the construction of your parser. The harder part of this, finding the FOLLOWS sets. However, you can make this task easy if you change the railroad diagrams to standard production notation.

**I would like to check this program, at the latest, on Monday before Thanksgiving break. That way, I would let you know what your current grades and you decide if you are happy with it or you want to take the final. You can always change your mind about taking the final. Remember that I could still see it on Tuesday, but I have other classes that day too. So, try to get it as early as possible. It is easier than what you think.**

*I would need the names of the students making up the groups so I can assign to each one of your grade.*