UNIVERSIDAD EAFIT	**EAFIT University** **School of Applied Sciences and Engineering** **Systems Engineering** **2025**
Subject:	**Formal Languages** **C2566-SI2002-5464 y C2566-SI2002-5465**
Professor:	**Adolfo Andrés Castro Sánchez**

Final Assignment

1. Deadline and Assessment

This is a group assignment, it is allowed to work in groups of two or three students. It is not allowed to work individually. All groups will give an oral presentation to defend their work. All members of the group should be present for the presentation.

Information about deadline, assessment and oral presentation are available at the course website or will be given in each group.

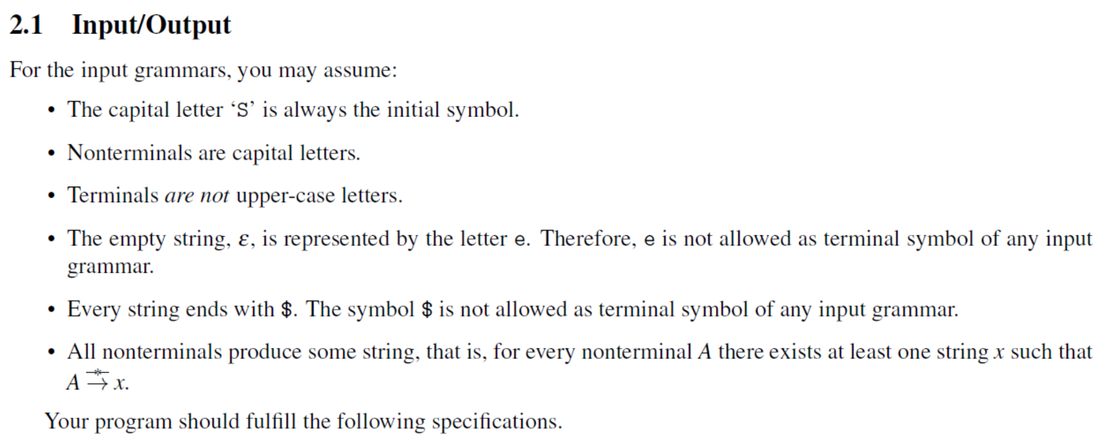
2. Assignment

For this assignment, see to Aho et al., Compilers: Principles, Techniques, and Tools (2nd Edition), sections 4.4 and 4.6.

The assignment consists of the following parts.

- Design and implement algorithms to compute the First and Follow sets (10%).
- Implement the algorithms to compute the LL(1) (Top-Down) and SLR(1) (Bottom-Up) parsers (20%).
- Prepare an oral presentation. All members of the group should be able to answer any questions about the submission (70%).

The algorithms can be implemented in any programming language.



2.1 Input/Output

For the input grammars, you may assume:

- The capital letter ‘`S`’ is always the initial symbol.
- Nonterminals are capital letters.
- Terminals *are not* upper-case letters.
- The empty string, ε, is represented by the letter `e`. Therefore, `e` is not allowed as terminal symbol of any input grammar.
- Every string ends with `$`. The symbol `$` is not allowed as terminal symbol of any input grammar.
- All nonterminals produce some string, that is, for every nonterminal A there exists at least one string x such that $A \overset{*}{\Rightarrow} x$.

Your program should fulfill the following specifications.

2.1.1 Input

The input is a context-free grammar. The input of the program is as follows.

- A line with a number $n > 0$ representing the number of nonterminals of the grammar to be analyzed.
- Then, your program should read n lines with the productions given in the following format:

```
<nonterminal> -> <derivation alternatives of the nonterminal
                  separated by blank spaces>
```

2.1.2 Output

Compute both parsers $LL(1)$ and $SLR(1)$, provided they satisfy the required conditions for each case.

2.1.3 Usage

One will specify which parser to use (if any). Your program should do as follows.

Cases.

1. The grammar is both $SLR(1)$ and $LL(1)$. Print

   ```
   Select a parser (T: for LL(1), B: for SLR(1), Q: quit):
   ```

   then it should receive strings to parse until an empty line is given. Then, print the same query as at the beginning.
2. The grammar is $LL(1)$, not $SLR(1)$. Print

   ```
   Grammar is LL(1).
   ```

   then it should receive strings to parse. Stop when the input is an empty line.
3. The grammar is $SLR(1)$, not $LL(1)$. Print

   ```
   Grammar is SLR(1).
   ```

   then it should receive strings to parse. Stop when the input is an empty line.
4. The grammar is neither $LL(1)$ nor $SLR(1)$. Print

   ```
   Grammar is neither LL(1) nor SLR(1).
   ```

For cases 1, 2 and 3, print in a single line `yes` when the a given string is generated by the grammar and `no` otherwise.

3. Assignment Submission

The activity will be submitted during the 17th week of classes. Please be creative and resourceful when creating the program's input and output, as well as the development.