UNITS 2, 3 AND 4: LEXICAL ANALYSIS AND GRAMMAR DESIGN FOR THE SYNTAX ANALYSIS

We want to construct a compiler for a language of definition and application of sequential machines. In the language you can define as many machines and process as many strings as you want. The language format is as follows:

- To declare a sequential machine, the MS instruction is used:
  ```
  MS name_of_the_sequential_machine
  {
  inputs { symbol_1, symbol_2,…,symbol_n}
  outputs { symbol_1, symbol_2,…,symbol_k}
  states { state_1, state_2, …}
  transitions {
  (state_1, symbol_input_1, state_2, symbol_output_1)
  (state_1, symbol_input_1, state_2, symbol_output_1)
  …
  (state_1, symbol_input_1, state_2, symbol_output_1)
  }
  }
  ```

- For the sequential machine to process a string, it is used:

  ```
  process ( name_of_automaton, string, initial_state)
  ```

The name of the sequential machine is a string of alphabetic characters. The statement of the set of states, the set of input and output symbols, and the set of transitions can be in any order, but they must always be included in every statement. The set of states, transitions, input and output symbols must have at least one value. An example of a sequential machine definition in this language would be:

```
MS Afirst
{
outputs {1,0}
states {a, c}
inputs {L,M,N}
transitions { (a,L,a,1)
(a,M,a,1)
(a,N,c,0)
(c,L,a,0)
(c,M,a,0)
(c,N,c,1)
}
}
process (Afirst, LLM, a)
process (Afirst, LLM, c)
```

To use the process function, the sequential machine used must be previously declared. The function displays, for the previous example, the following information:

```
MS: Afirst    Input: LLM  Output: 111
MS: Afirst    Input: LLM  Output: 011
```

It is required:

1. Define the grammar G that would generate valid sentences of this programming language.

SOLUTION:

A grammar that generates the language of the problem is defined as follows:

G = {**A**, **B**, **D**, **E**, **R**, **S**, **V**, **W**, **Z** }, {( ) , { } automatonFD string states final initial name recognize t transitions}, {**S**}

(1) **A**::= λ
(2) **A**::= **S**
(3) **B**::= states { **V** }
(4) **B**::= final { **U** }
(5) **B**::= initial { t }
(6) **B**::= transitions { **W** }
(7) **D**::= automatonFD name { **B B B B** }
(8) **E**::= **D**
(9) **E**::= **R**
(10) **R**::= recognize (name, string)
(11) **S**::= **E A**
(12) **U**::= λ
(13) **U**::= **V**
(14) **V**::= t **Z**
(15) **W**::= λ
(16) **W**::= (t , t , t) **W**
(17) **Z**::= λ
(18) **Z**::= , **V**