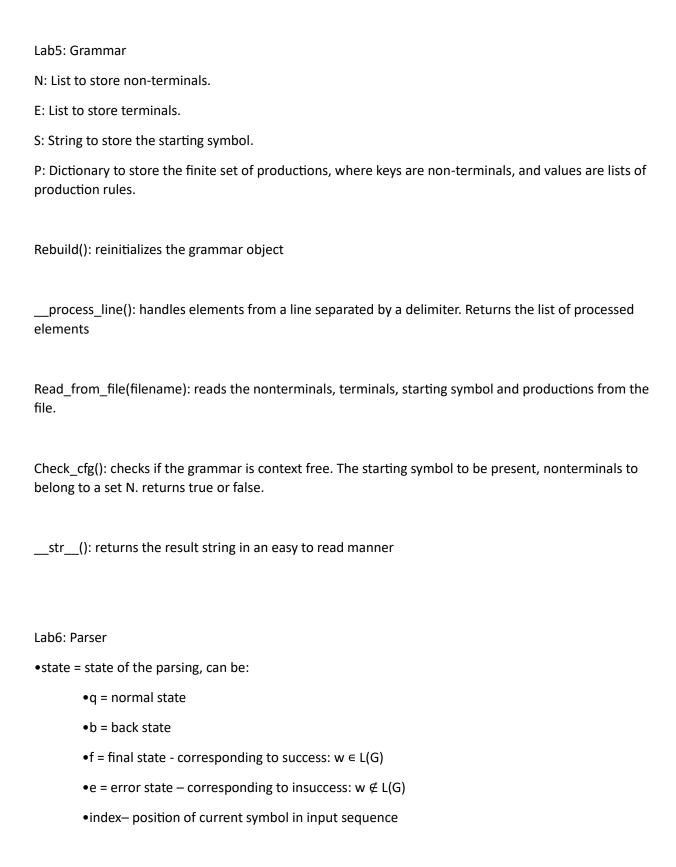
https://github.com/911-Abrudan-Rebeca/FLCD/tree/main/Lab5



 $w = a1a2...an, i \in \{1,...,n+1\}$

•working = working stack, stores the way the parse is built

•input = input stack, part of the tree to be built

Read_sequence(seq_file): function that reads a sequence from a file and returns it as a list

Get_situation(): returns the current: state, index, working stack, input symbol

Expand(): head of input stack is a nonterminal

removes first symbol from input (nont), pushes it onto the w stack as a tuple (prod index=0), retrieves first production for the nont from the grammar, adds to the existing input

advance():head of input stack is a terminal = current symbol from input

removes the first symbol from the input and pushes it onto the w stack, increments the index

momentary_insuccess(): head of input stack is a terminal ≠ current symbol from input, momentary failure => b

back(): head of working stack is a terminal

pops item from working stack and inserts it at the beginning of the input, decrements index

success(): state is f

another_try(): head of working stack is a nonterminal

nont => tries another production for the nont, if there is another prod => updates the working stack and input, no prod => adjust the input based on the length of the last prod/trans to error state

error(): state is e

run(): algorithm for descendent recursive like in the course. Continues until final state==f or e.

Lab7: ParserOutput and parsing table

Class Node: integer index for the father and sibling, a value for the node, integer index for the production rule

__str__(): returns the string in an easy to read manner. It displays the value, father index and sibling index.

Class ParserOutput: the grammar, the sequence (used to generate the parsing tree), the tree (a list which stores the node)

Read_sequence(sequence): reads the input from a file and returns a list of strings.

Create_parsing_tree(working): iterates through the working stack and creates a node instance for each element. It also establishes relationships between nodes.

Get_len_depth(index, working): computes the length and depth of the production rule in the parsing tree

Write_parsing_tree(state, working, output_file): prints the parsing tree in a table format