https://github.com/dragos231456/LFCD/tree/master/5

In this lab we have implemented the recursive descent algorithm.

By using the grammar.py file, we can check what the terminals, non-terminals, start symbol and productions are, check if the sequence is a CFG and also in the sequence is accepted.

By using the RDParser.py file, we can parse a sequence/program and check if it's valid. If it is, we are also displaying the parse tree.

Input:

- the input files g1.txt and g2.txt have the following strucuture:

- 1st line: space separated non-terminals

- 2nd line: space seuparated terminals

- 3rd line: start symbol

- following lines: productions

Grammar:

- this class is used to read the grammar from the file and it holds the structure described above.

State:

- this is the class used to represent the recursive descent state, so it stores the following:

- the state type (NORMAL, ERROR, BACK, FINAL)

- the current index from the input sequence

- the work_stack -> each element is a tuple (a, b), where a is a symbol (terminal or nonterminnal) and b is the index of the current production

- the input_stack -> same structure as the work_stack

RDParser:

- this is the class where the recursive descent algorithm is used. It has several functions related to the algorithm (advance, expand, momentary_insuccess, back, another_try, success) and also the method in which we compute the parse tree.

- it uses the State class described above to keep track of all the variables involved in the algorithm