https://github.com/BrejeMihai/FLCD_Lab08

Documentation available also on the repository

```
class Grammar(object):

    def __init__(self):
        self.terminals = []
        self.non_terminals = []
        self.productions = defaultdict(list)
        self.starting_symbol = ""

    def __str__(self):
        pass

    """
    summary: Gets all productions for a specified non_terminal

    :param non_terminal: string corresponding to an existing non_terminal

    :return: a list of all the productions of the given non_terminal
    """
    def get_productions_for_nonterminal(self, non_terminal):
        pass

    """
    summary: Reads a grammar from a file

    :param path: path to the grammar file

    :return: void, modified Grammar object
    """
    def read_from_file(self,path):
        pass
```

```python
class Lr0Parser(object):

    def __init__(self, grammar):
        self.grammar = grammar
        self.terminals = grammar.terminals
        self.non_terminals = grammar.non_terminals
        self.productions = grammar.productions
        self.starting_symbol = grammar.starting_symbol
        self.augmented_grammar = None
        self.dotted_productions = None

    """
    summary: Entry point of the parser, augments grammar and creates the initial closure
    """
    def start_magic(self):
        pass

    """
    summary: Augments the grammar already present in the object
    :return: void, modified attribute "dotted_productions" in object
    """
    def augment_grammar(self):
        pass

    """
    summary: Transforms the productions from a map representation into a list representation, which can be more easily /
            parsed for the reduce indexing in the parsing table
    :return: void, modified attribute "indexable_productions" in object
    """
    def indexable_productions(self):
        pass

    """
    summary: Implementation of the closure function, will call itself recursively where necessary

    :param element: a given transition
    :param closure_history: map containing the shifted transitions corresponding to this call of closure
    :param transition_history: map containing the already done transitions
    """
    def closure(self, element, closure_history, transition_history):
        pass
```

```
    """
    summary: Shifting of the dot in a given transition

    :raises Exception: when dot is already at the last index

    :param transition_ref: transition to be shifted (immutable behaviour, copy will be made)

    :return string: the shifted transition
    """
    def shift_dot(self, transition_ref):
        pass

    """
    summary: Implementation of the goto function

    :param initial_transition: the initial transition
    :param key: parent "key" of the transition given
    :param state: state at the moment of the goto
    :param parent: transition corresponding to the parent key

    :return: void, modified attribute "queue" of the object
    """
    def goto(self, initial_transition, key, state, parent=-1):
        pass

    """
    summary: checks for collision before replacing a value in the map

    :param value: the value already existing in the parsing table
    :param new_value: the new value which to be inserted
    :param where: location in the parsing table

    :raises Exception: when collision occurs
    """
    def check_for_collision(self, value, where, new_value):
        pass
```

```
    """
    summary: Function which will call goto for all the next states, creates new states and populate the parsing /
            table, also checks for collision

    :param state: the beginning state
    :param initial_dotted: the augmented grammar at first, the shifted transition after
    :param parent: the parent "key"
    :param parent_transition: transition corresponding to the parent key

    :return void, modified attribute "state" and "inner_table_values" of the object
    """
    def goto_all(self, state, initial_dotted, parent=-1, parent_transition="-1"):
        pass

    """
    summary: Gets all the states that are reduced

    :return list: a list of all reduced states
    """
    def get_reduced(self):
        pass

    """
    summary: Implementation of the canonical collection function

    :return void, modified attributes "inner_table_values", "queue", "state_parents", "states" of the object
    """
    def canonical_collection(self):
        pass

    """
    summary: Util function to print the parsing table

    :return void, modified stdout and file
    """
    def show_parsing_table(self):
        pass
```

```
    """
    summary: Util function to print to file

    :param string: the string to print
    :param file_handle: the file handle of the corresponding file

    :return void, modified file
    """
    def print_and_write_to_file(self, string, file_handle):
        pass


    """
    summary: Function implementing the parsing algorithm of a sequence

    :param word: list of tokens (sequence to be parsed)

    :raises Exception: when the input sequence is not in list format
    :raises Exception: when the Reduce couldn't be completed with the actual stacks

    :return void, modified stdout and file
    """
    def actual_parsing(self, word):
        pass


    """
    summary: Util function to print a map in a more formatted way

    :param map: the map to be printed
    :param message: a message to be printed beforehands

    :return string, corresponding to the formatted string of the map
    """
    def pretty_print_map(self, map, message=None):
        pass
```