https://github.com/DragosteSergiu/Facultate/blob/main/Lab6FLCD.py

class Action is used in representation of the field 'action_table' from class LR0Parser.
- It contains two fields, which are 'name', and 'operand'.
- The field 'name' is used for storing the name of the action. It can be 'ACCEPT', 'REDUCE', 'SHIFT'.
- The field 'operand' is used for 'REDUCE' and 'SHIFT' actions. For 'REDUCE' action, it contains the number of production. For 'SHIFT' action, it contains the number of the state in which the program will shift.
- It contains getters and setters for both fields.
- It overrides the '__str__()' method.

class LR0Item extends the class Rule and it is used to represent a dot production.
- It has a field 'dot_position' which is represented as an integer and stores the position of the '.' in production.
- It has the field 'left_side' and 'right_side' of the parent class.
- It overrides the getters and setters of the parent class, and has getters and setters for the field 'dot_position'.
- It overrides the method which adds an element to the field 'right_side'.
- It has a method which returns the element from the field 'right_side' from position 'dot_position'.
- The class overrides the methods '__str__()' and '__eq__()' of the parent class.
- It also overrides the method '__hash__()' it order to be a hashable object.

Class LR0State is used to represent a state of the LR0Parser.
- It has three fields which are 'number', 'items', 'transitions'
- The field 'number' is used to recognize each state.
- The field 'items' is represented as a list of LR0Items and it is used to store all items of a state.
- The field 'transitions' is represented as a map with a tuple (LR0Items, terminal/non-terminal) as key and a LR0State as value.
- It has getters and setters for its fields.
- It has the method 'add_items()' which adds an item to the field 'items'.
- It has the method 'add_transition()' which adds an element to the field 'transitions'.
- It has the method 'update_transition()' which updates an element from the field 'transitions'.
- The class overrides the method '__str__()' and '__eq__()'.
- The class implements the method 'compare_items()' which compares the field 'items' with a list given as parameter.

Class LR0Parser is used to represent the actual parser.
- It has four fields which are 'grammar', 'canonical_collection', 'action_table' and 'go_to_table'.
- The field 'grammar' stores an object of type Grammar.
- The field 'canonical_collection' is represented as a list of LR0State objects and it is used to store all states of the grammar.
- The field 'action_table' is represented as a map having as key a tuple (number_of_state, terminal) and as value an Action object.
- The field 'go_to_table' is represented as a map having as key

a tuple (number_of_state, non-terminal) and as value the number of the next state.
- The method 'enhance_grammar()' is used to add a new rule in the grammar.
- The method 'closure()' takes as parameter an object of type LR0Item and returns a list of object of type LR0Item.
- The method 'goTo()' takes as parameter an object of type LR0Item and returns another object of type LR0Item incrementing the field 'dot_position'.
- It implements the method 'create_canonical_collection()' which creates the states that are stored in the field 'canonical_collection'.
- It implements the method 'create_action_table()' which creates the map that contains information about transitions which contain an terminal on the second position of the key.
- It implements the method 'create_go_to_table()' which creates the map that contains information about transitions which contain an non-terminal on the second position of the key.
- It contains the method 'canonical_collection_to_string()' which returns the value of field 'canonical_collection' as a string.
- It contains the method 'canonical_action_table_to_string()' which returns the value of field 'canonical_collection' as a string.
- It contains the method 'canonical_go_to_table_string()' which returns the value of field 'canonical_collection' as a string.
- The class implements the method 'accept()', which verifies if a given sequence of characters is accepted by the parser and returns the list of productions that must be used in order to obtain the given sequence.