https://github.com/eduardpauliuc/flcd

Symbol table documentation:

Methods:
- initialisation: SymbolTable()
- hash(value)
    - Static method that accepts an integer, string or character and computes the hash value of that value
- add(value)
    - Return value is a position in the table, a tuple of two integers, first is the hash bucket the value should belong in and the second one is the position in that bucket, which is implemented using arrays.
    - If value was already present in the SymbolTable, it returns the position at which it is currently stored
    - If the value is not present, it is added to the corresponding bucket and returns the new position.

The MOD value for the hash function is a static variable in the class and it should be a prime number. It also determines the number of buckets in the symbol table.

Scanner:
- has constants_table of type SymbolTable
- has identifiers_table of type SymbolTable
- scan method does the scan and then creates the output ST and PIF files
- throws lexical error if needed

Regex expressions used:
- for string constants: ^\"([a-zA-Z0-9_+\-*/%<=>!:,]*)\”
    - strings like: “”, “text”, “a9_+=“
- for number constant: ^(([+-]?[1-9]+[0-9]*)|0)
    - signed/unsigned numbers: 0, 100, -1233, +121211
- for identifiers: ^\$[_a-zA-Z]+[_a-zA-Z0-9]*
    - identifiers: $_123, $aA231
    - must start with a $ sign and then something that si not a number

PIF structure:

- list with tuples [value, position]. On the first position we have ‘id’ if an identifier was added, ‘const’ if a constant was added and the value of the token if a token was identified.
- on the second position, we have a position from identifiers table in case of ‘id’, position from constants in case of ‘const’ and -1 in case of token.

Automata file structure:

```
<automata_file> ::= <states_line> <alphabet_line> <initial_state_line> <final_states_line>
                    <transitions_lines>

<states_line> ::= <states_line> <state> | <state>
<alphabet_line> ::= <alphabet_line> <element> | <element>
<initial_state_line> ::= <state>
<final_states_line> ::= <final_states_line> <state> | <state>
<transitions_lines> ::= <transition_lines> <transition_line> | <transition_line>
<transition_line> ::= <state> | <transition_line> <element> <state>
```