**Lab 4 (part 1)**

<https://github.com/cinnamonbreakfast/flcd/tree/main/lab4_1>

*Data represented on:*

**States**: List<String>

**Alphabet**: List<String>

**Transitions**: Dict<String, List<String>> # some sort of key: pair

**Final** **states**: List<String>

*(Using functional programming, no classes)*

**Predefined tokens are emphasized.**

Program ::= **entry** cmpdstmt **;**

Type ::= **bool** | **int** | **char** | **string**

Assignstmt ::= IDENTIFIER **=** expression

decl ::= declstmt | declasgnstmt

Declstmt ::= type IDENTIFIER

Declasgnstmt ::= type IDENTIFIER **=** expression

Cmpdstmt ::= **{** stmtlist **}**

Stmtlist ::= stmt | stmt **;** stmtlist

Stmt ::= simplstmt | structstmt

Simplstmt ::= assignstmt | decl | iostmt

Iostmt ::= **INPUT(** IDENTIFIER **) | WRITE(** IDENTIFIER **)**

Value ::= integer\_const | character | string\_const | IDENTIFIER | arrayAccess | expression

Expression ::= value arithmetic\_ops **(** expression **)**

Arithmetic\_ops ::= + | - | / | \*

Term ::= term **\*** factor | factor | arrayAccess

Factor ::= **(** expression **)** | IDENTIFIER

Structstmt ::= cmpdstmt | ifstmt | whilestmt

Whilestmt ::= **while (** condition **)** cmpdstmt

Ifstmt:**:= if(** CONDITION **)** cmpdstmt **else** cmpdstmt

Condition ::= expression RELATION expression

Relation ::= **<** | **<=** | **==** | **>=** | **===** | **>**

Arraydecl ::= type IDENTIFIER **[** number **]**

arrayAccess::= IDENTIFIER[ IDENTIFIER ]

Mapdecl ::= **map{** type : type **}**

mapAccess ::= IDENTIFIER[ IDENTIFIER ]

**TOKEN LIST:**

+ - \* / := < <= = >= === ~ % & ^ **array, map, const, do, else, if, int, elif while, for, range, class, struct, string, float, char, boolean, input, print, return, fun, key, value, main, entry**