Lisp part:

Algorithm:

A global list containing Context Free Grammar rules has been defined. Lisp functions from the previous assignment were used to check whether the input follows this grammar rule or not. In the previous assignment, the input was divided into individual words and these words were recognised if they were keywords. In this assignment, valuef and identifier were also recognised besides keywords. The inputs that are separated into parts are converted into tokens and a list is thrown. If the lists consisting of these tokens are mapped with any CFG rule, the process starts. Before mapping with CFG rule, the expressions in the expression are also simplified. In this way, it can be understood whether any CFG rule is followed. If a function is suitable for a CFG that creates a function, the function id and the expression part are saved in a hash-table. If it is an expression, it enters a large function called evalution, where the function call sends the expression matching the id back to this function and the result is calculated. If it is an expression other than a function call, it is checked whether there is an expression in the expression and the result is calculated.

Others:

Reads from the file. Reading from the terminal. Exits with the (exit) command. Exits if there is a syntax error.

Yacc part:

Algorithm:

CFG rules are set. A c struct has been created for valuef. If it realises that an input comes from lexer in valuef format, it is converted into a valuef object with the valuefCreator function and sent to the yacc file. In the yacc file, it analyses according to CFG rules and sends it to addition, subtraction, multiplication and division functions.

Others:

Reads from the file. Reading from the terminal. Exits with the (exit) command. Exits if there is a syntax error.