

# CSE 341 PROGRAMMING LANGUAGES FALL 2020

## HOMEWORK 3 REPORT

**Emirhan UZUN**

**171044019**

In this homework, we are asked to implement the syntax analysis in two different ways:

1. There are tools to implement syntax analysis given the rules in a meta-grammar such as CFGs. One such tool is “Yacc” that we generate C code to do the syntax analysis.
2. For this course, we will be implementing a syntax analysis algorithm for G++ in Lisp as well. For this we are not expected to use a meta-grammar to define the lexical syntax of your language.

### **PART1 (YACC)**

In this part, I edited my gpp\_lexer.l source code firstly. I used in previous homework. I added some tokens and also I edited the filename, value and identifier tokens rules.

I did a makefile for this part. Because we have to compile more file. So you can write “make” to the terminal and then you can execute the program like

**1 - ./a.out or;**

**2 - ./a.out “filename”**

The first one is read the inputs from terminal until you enter “(exit)” and the second one is read the input from file and it closes the program when it come to end of the file.

**NOTE ! : I print the values for float in operator tokens (+ , - \* , / , \*\*). Because for instance, if you find the value of (/ 4 6), then program will print 0. But I didn't want that so now it will print 0,67.**

## SCREENSHOTS

```
emir@emir:~/Desktop/bittik/hw3/part1$ make clean
rm a.out
rm lex.yy.c
rm y.tab.c
rm y.tab.h
emir@emir:~/Desktop/bittik/hw3/part1$ make
lex gpp_lexer.l
yacc -d gpp_interpreter.y
gcc lex.yy.c y.tab.c -w
emir@emir:~/Desktop/bittik/hw3/part1$ ./a.out
$ ( + 5 6)
11.00
$ (- 66 44)
22.00
$ (/ 54 4)
13.50
$ ( * 23 7)
161.00
$ (** 9 3)
729.00
$ (list 44 5632 65)
(44 5632 65)
$ (set x 2)
2.00
$ x
2.00
$ (set a (append 77 (list 88 10 11 34)))
(77 88 10 11 34)
$ (defvar b (list 14 76 23))
b
$ (disp (+ 222 346))
568.00
$ (if (and true true) (list 11 22 33))
(11 22 33)
$ (defun myFun (y) (+ 5 3))
myfun
$ (not true)
FALSE
$ (not false)
TRUE
$ (not (and true false))
TRUE
$ (equal 24 68)
FALSE
$ (equal 16 16)
TRUE
$ (less 58 50)
FALSE
$ (less 48 59)
TRUE
$ (exit)
Program closed.
emir@emir:~/Desktop/bittik/hw3/part1$
```

In this part, first I clean the compiled code and then again I make the files. After that if you execute the program like

- “ ./a.out”

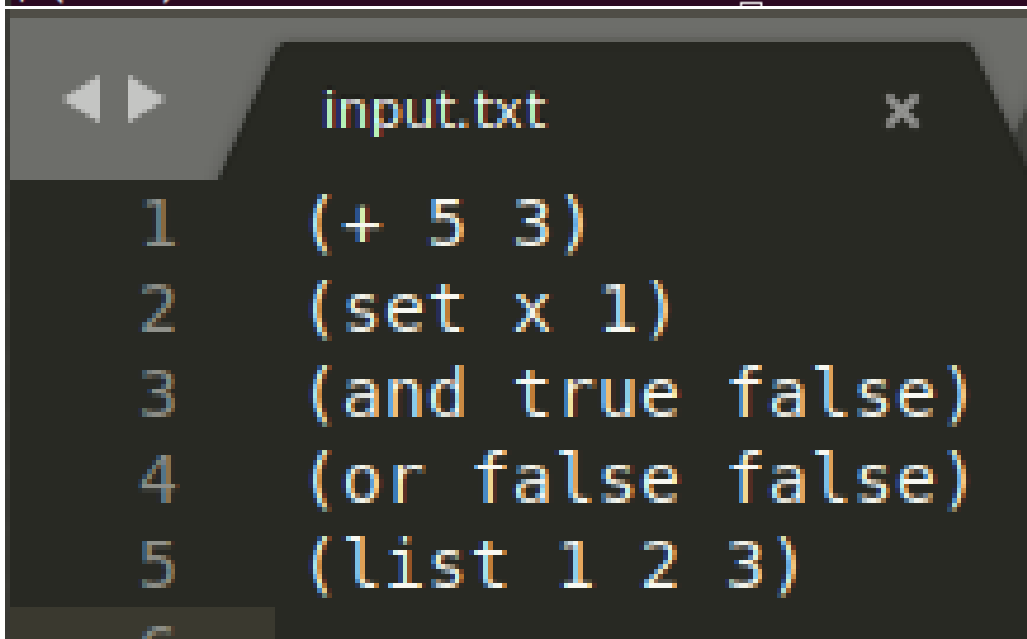
then it will read the input from terminal.

**NOTE ! : I set the one variable but I couldn't like this example :**

(+ x y)

**So my program doesn't hold the two variable. It just hold one variable.**

```
emir@emir:~/Desktop/bittik/hw3/part1$ make clean
rm a.out
rm lex.yy.c
rm y.tab.c
rm y.tab.h
emir@emir:~/Desktop/bittik/hw3/part1$ make
lex gpp_lexer.l
yacc -d gpp_interpreter.y
gcc lex.yy.c y.tab.c -w
emir@emir:~/Desktop/bittik/hw3/part1$ ./a.out "input.txt"
$ 8.00
$ 1.00
$ FALSE
$ FALSE
$ (1 2 3)
```



The screenshot shows a text editor window with the title 'input.txt'. It contains five lines of code, each preceded by a line number from 1 to 5. The code is as follows:

Line	Code
1	(+ 5 3)
2	(set x 1)
3	(and true false)
4	(or false false)
5	(list 1 2 3)

If you want to read from file, you can enter the terminal  
“./a.out filename ”

## PART2 (LISP)

We have to implement our interpreter in Common Lisp. Hand in our “gpp\_interpreter.lisp” file for this part of the homework.

You can execute the program like

**1 – clisp gpp\_interpreter.lisp or**

**2 - clisp gpp\_interpreter.lisp “filename”**

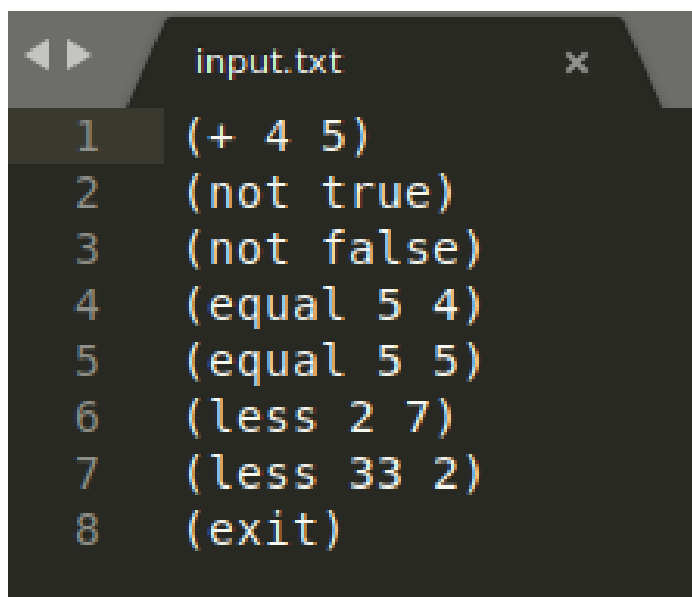
**NOTE ! : I couldn't do EXPLISTI part. And the operators + and -, doesn't print the right values for 2 digits and more. I could do +, -, \*, /, \*\*, and, or, less, equal , not and exit. There is no other parts.**

## SCREENSHOTS

```
emir@emir:~/Desktop/bittik/hw3/part2$ clisp gpp_interpreter.lisp
$ (+ 6 1)
7
$ (- 5 3)
2
$ (/ 9 4)
9/4
$ (* 3 3)
9
$ (** 5 2)
25
$ (and true false)
NIL
$ (and true true)
T
$ (or true false)
T
$ (not true)
NIL
$ (not false)
T
$ (disp (+ 5 3))
8
8
$ (equal 11 11)
T
$ (less 25 46)
T
$ (less 110 22)
T
$ (exit)
"Program Closed."
```

Without filename, you can execute the program like that.  
**“clisp gpp\_interpreter.lisp”**

```
emir@emir:~/Desktop/bittik/hw3/part2$ clisp gpp_interpreter.lisp "input.txt"
9
NIL
T
NIL
T
T
NIL
"Program Closed."
emir@emir:~/Desktop/bittik/hw3/part2$
```



The image shows a text editor window with the title 'input.txt'. It contains a list of 8 lines of code, each preceded by a line number from 1 to 8. The code is written in a Lisp dialect and includes arithmetic operations, logical tests, and a program termination command.

Line	Code
1	(+ 4 5)
2	(not true)
3	(not false)
4	(equal 5 4)
5	(equal 5 5)
6	(less 2 7)
7	(less 33 2)
8	(exit)

If you want to execute the program with filename , you can write:  
**clisp gpp\_interpreter.lisp "filename"**