

CSE341 – Programming Languages (Fall 2020)

Homework #3 Report

Berk PEKGÖZ
171044041

All syntax rules given in GppSyntax.pdf file (Green = I could implement, Red = I could not implement)

- Comment >; ...
- Listvalue > (list ...)
- Add operation > (+ ...)
- Sub operation > (- ...)
- Mult operation > (* ...)
- Div operation > (/ ...)
- Dbmult operation > (** ...)
- Identifier
- Value
- And operation > (and ...)
- Or operation > (or ...)
- Not operation > (not ...)
- Equal operation (with EXPB) > (equal ...)
- Equal operation (with EXPI) > (equal ...)
- BinaryValue > true/false
- Assignment operation > (set ...)
- **Function definition and function call > (deffun ...) / (id)**
- If operation (with EXPI) > (if ...)
- If operation (with EXPLISTI) > (if ...)
- **For operation > (for ...)**
- Concat operation > (concat ...)
- Append operation > (append ...)

Note: Some keywords do not have rule (like “sumup”) and some rules do not have keyword (like “while”). Mr. Dede said we do not have to implement them in Q&A session. So, I did not mention them above.

Some Explanations About Implemented Operations

If operation: In GppSyntax.pdf file there are two types of “if”. EXPI -> (if EXPB EXPLISTI) and EXPI -> (if EXPB EXPLISTI EXPLISTI). Mr. Dede announced on Moodle that we must implement only EXPI -> (if EXPB EXPLISTI) [no else situation]. But there is a problem on this rule. This rule says if EXPB is true then implement EXPLISTI and return its value as EXPI. This is not possible. For example, (list 1 2 3) must return a list (1 2 3) but (1 2 3) is not an EXPI. So, I changed and implemented both versions of EXPI -> (if EXPB EXPLISTI) as **EXPI -> (if EXPB EXPI) and EXPLISTI -> (if EXPB EXPLISTI)**.

Also, there is no else part in this type of “if” so if EXPB is false then it returns 0 or (0).

Listvalue: ‘(...)’ in GppSyntax.pdf file but we don’t have ‘ operator so it is implemented as (list ...).

Identifier: If an identifier is not assigned before, returns 0. For example, if program does not know the value of X then (+ X 5) returns 5.

Some Explanations About Unimplemented Operations

Deffun operation: In GppSyntax.pdf file there is a rule about this. EXPI -> (deffun Id IDLIST EXPLISTI). Some parts of this rule do not make sense. First of all, there is no rule about IDLIST in pdf file. Secondly, after IDLIST only EXPLISTI can come and this does not make sense because with this way deffun can be only this type of things:

- (deffun myFunc IDLIST (list 1 2 3))
- (deffun myFunc IDLIST (append 5 (list 1 2 3)))
- (deffun myFunc IDLIST (concat (list 4 5 6) (list 1 2 3)))

And these must return EXPI but any of these are not EXPI and deffun cannot be like (deffun myFunc IDLIST (+ 4 5)) so I thought this does not make sense and I did not implement it.

For operation: In GppSyntax.pdf file there is a rule about this. EXPI -> (for (Id EXPI EXPI) EXPLISTI). I think this rule does not make sense. Because only EXPLISTI can come after range specified, and this does not make sense because with this way “for” can be only this type of things:

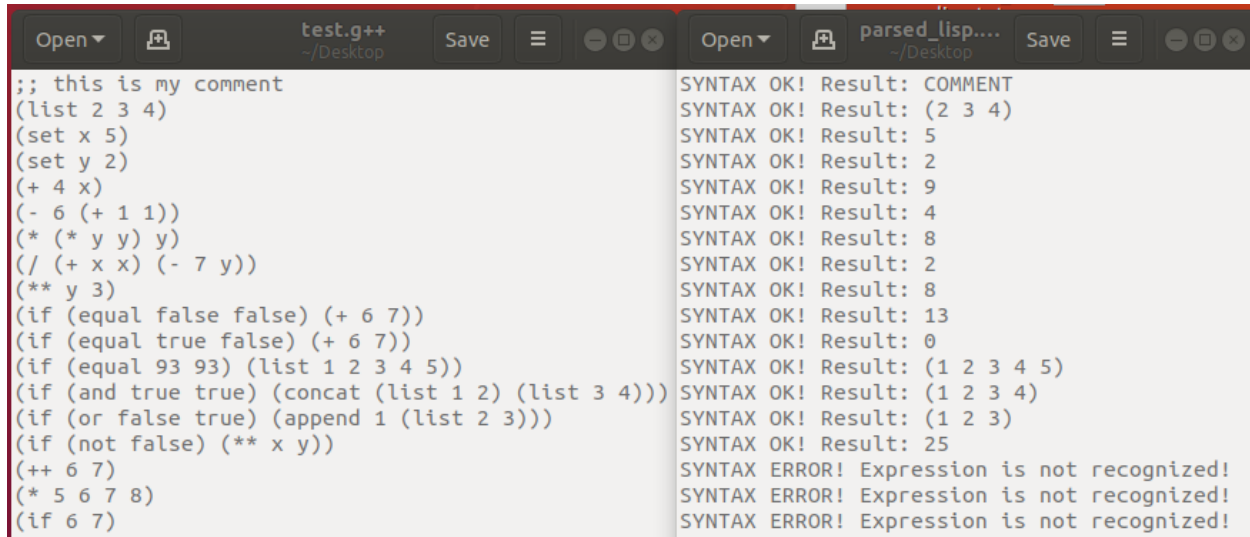
- (for (x 0 10) (list 1 2 3))
- (for (x 0 10) (append 5 (list 1 2 3)))
- (for (x 0 10) (concat (list 4 5 6) (list 1 2 3)))

And these must return EXPI but any of these are not EXPI and “for” cannot be like (for (x 0 10) (+ 4 5)) so I thought this does not make sense and I did not implement it.

Some Tests

Input file / Output file

Tested on Lisp parser.



test.g++	parsed_lisp....
; ; this is my comment	SYNTAX OK! Result: COMMENT
(list 2 3 4)	SYNTAX OK! Result: (2 3 4)
(set x 5)	SYNTAX OK! Result: 5
(set y 2)	SYNTAX OK! Result: 2
(+ 4 x)	SYNTAX OK! Result: 9
(- 6 (+ 1 1))	SYNTAX OK! Result: 4
(* (* y y) y)	SYNTAX OK! Result: 8
(/ (+ x x) (- 7 y))	SYNTAX OK! Result: 2
(** y 3)	SYNTAX OK! Result: 8
(if (equal false false) (+ 6 7))	SYNTAX OK! Result: 13
(if (equal true false) (+ 6 7))	SYNTAX OK! Result: 0
(if (equal 93 93) (list 1 2 3 4 5))	SYNTAX OK! Result: (1 2 3 4 5)
(if (and true true) (concat (list 1 2) (list 3 4)))	SYNTAX OK! Result: (1 2 3 4)
(if (or false true) (append 1 (list 2 3)))	SYNTAX OK! Result: (1 2 3)
(if (not false) (** x y))	SYNTAX OK! Result: 25
(++ 6 7)	SYNTAX ERROR! Expression is not recognized!
(* 5 6 7 8)	SYNTAX ERROR! Expression is not recognized!
(if 6 7)	SYNTAX ERROR! Expression is not recognized!

Note: In Yacc part if there is a syntax error program ends but I make lisp with this way to be easy to test. It can be changed with one flag.

Note 2: In Yacc part executable file is named "gpp_interpreter.out".

Note 3: Output text file for Yacc is named "parsed_cpp.txt" and Output text file for lisp is named "parsed_lisp.txt".