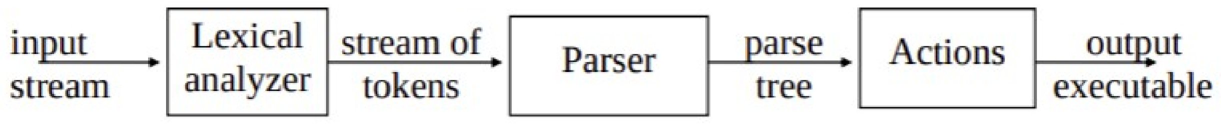
HW 3 REPORT

Can Duyar – 171044075

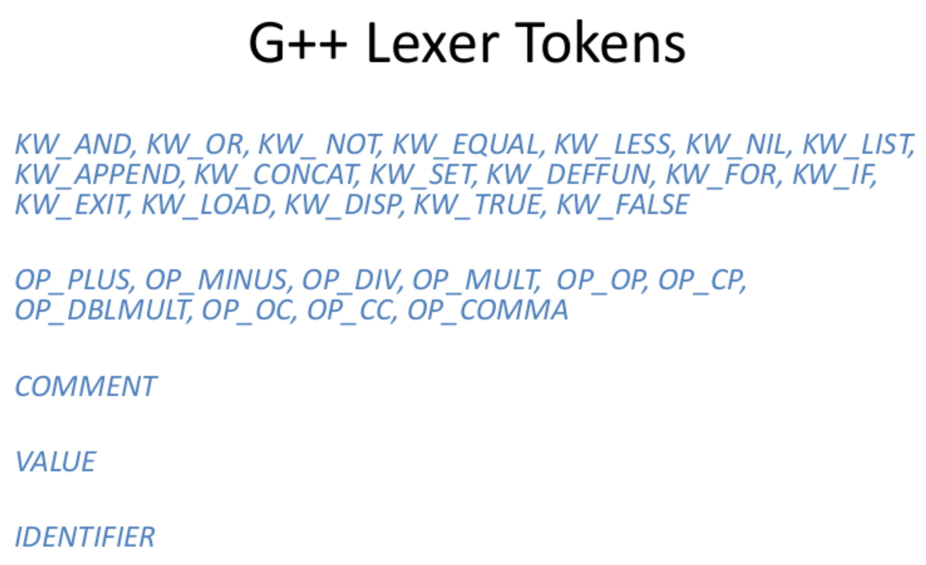
G++ Language Interpreter using Flex and Yacc: (All parts work correctly):

I implemented my interpreter using Yacc. I also used my previous lexer with some additional parts to design my interpreter correctly. My lexer gives tokens to my parser . This process known as lexical analyze, lexical analyzer scans the input stream and converts sequences of characters into tokens.

Main Logic:



Lexer Tokens of G++ programming language were provided by PDF that you sent.



G++ Lexer Tokens

KW_AND, KW_OR, KW_ NOT, KW_EQUAL, KW_LESS, KW_NIL, KW_LIST, KW_APPEND, KW_CONCAT, KW_SET, KW_DEFFUN, KW_FOR, KW_IF, KW_EXIT, KW_LOAD, KW_DISP, KW_TRUE, KW_FALSE

OP_PLUS, OP_MINUS, OP_DIV, OP_MULT, OP_OP, OP_CP, OP_DBLMULT, OP_OC, OP_CC, OP_COMMA

COMMENT

VALUE

IDENTIFIER

I also implemented a main function on my .l file to control terminal operations. There are two options for this part;

1)If there is an input after ./a.out then it reads from file that has name which is given from terminal. (./gpp_interpreter.out filename.txt **Note:** If occurs SYNTAX ERROR when it reads from the file then program will exit.)

2)if there is no input as a filename after ./a.out then it continuous as interpreter without a file. (./gpp_interpreter.out)

Important notes for my interpreter design: (from moodle-page instructions):
* I implemented assignment operation → (setq Id EXPI)
* Implement function definition and the body (deffun)
* In the if statement I implemented just the first case **EXPI -> (if EXPB EXPLISTI)**

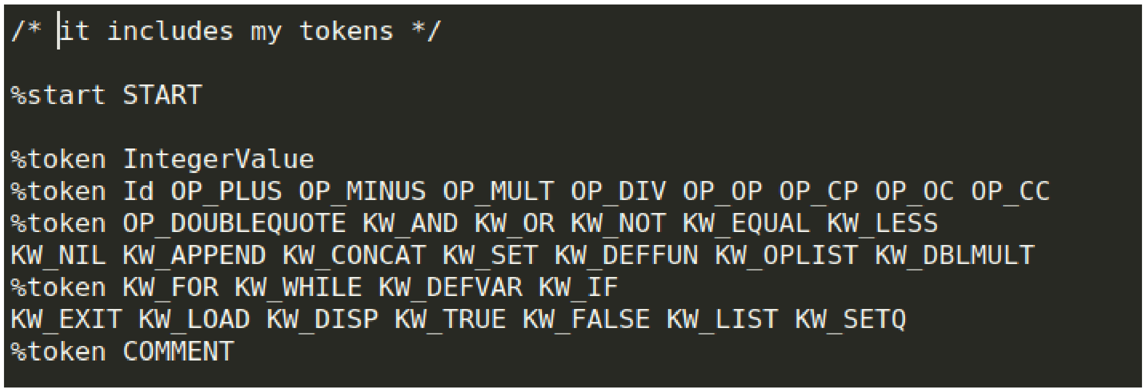
Design of my yacc file (gpp_interpreter.y)

→ I used 2 different functions known as *operationAppend* and *showArray* on my .y file.

OperationAppend → it takes a single-array,size of the array and a number as formal parameters. I used this function when I implemented KW_APPEND operation.

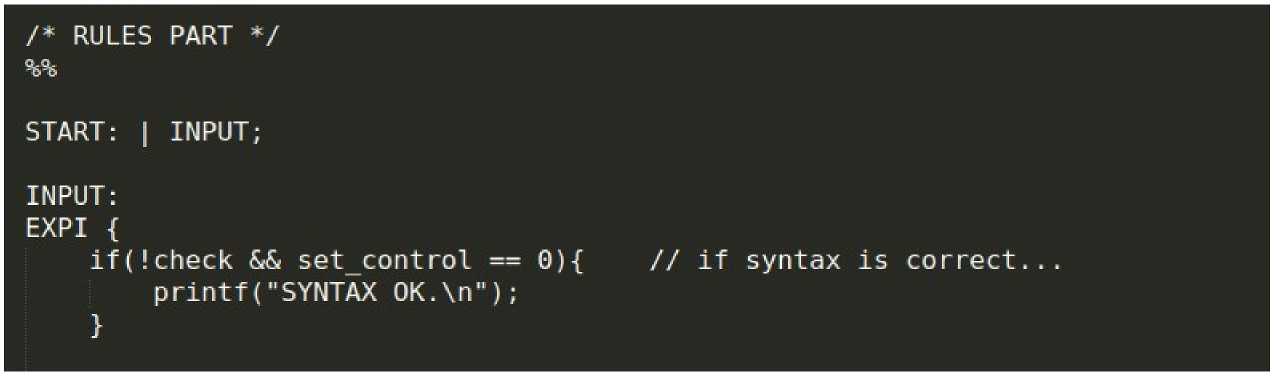
ShowArray → It prints the values of an array. When I want to print the Result values of the operations(for example in KW_LIST operation etc.), I used it.

My gpp_interpreter.y code has tokens that come from my lexer(gpp_interpreter.l)



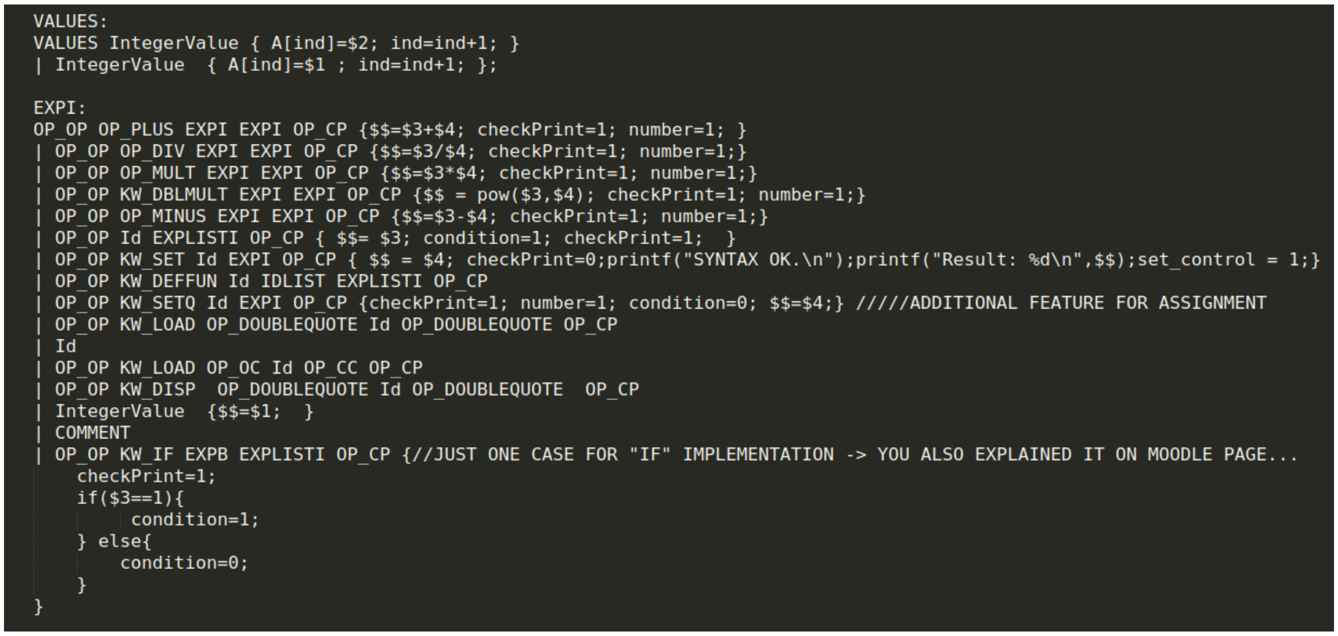
```
/* it includes my tokens */

%start START

%token IntegerValue
%token Id OP_PLUS OP_MINUS OP_MULT OP_DIV OP_OP OP_CP OP_OC OP_CC
%token OP_DOUBLEQUOTE KW_AND KW_OR KW_NOT KW_EQUAL KW_LESS
KW_NIL KW_APPEND KW_CONCAT KW_SET KW_DEFFUN KW_OPLIST KW_DBLMULT
%token KW_FOR KW_WHILE KW_DEFVAR KW_IF
KW_EXIT KW_LOAD KW_DISP KW_TRUE KW_FALSE KW_LIST KW_SETQ
%token COMMENT
```

gpp_interpreter.y also has a "RULES PART " → it uses my tokens and creates the parse tree for my G++ language. Tokens are used in this part according to Concrete Syntax Rules that are given on the PDF.



```
/* RULES PART */
%%

START: | INPUT;

INPUT:
EXPI {
    if(!check && set_control == 0){    // if syntax is correct...
        printf("SYNTAX OK.\n");
    }
```

.
.
.



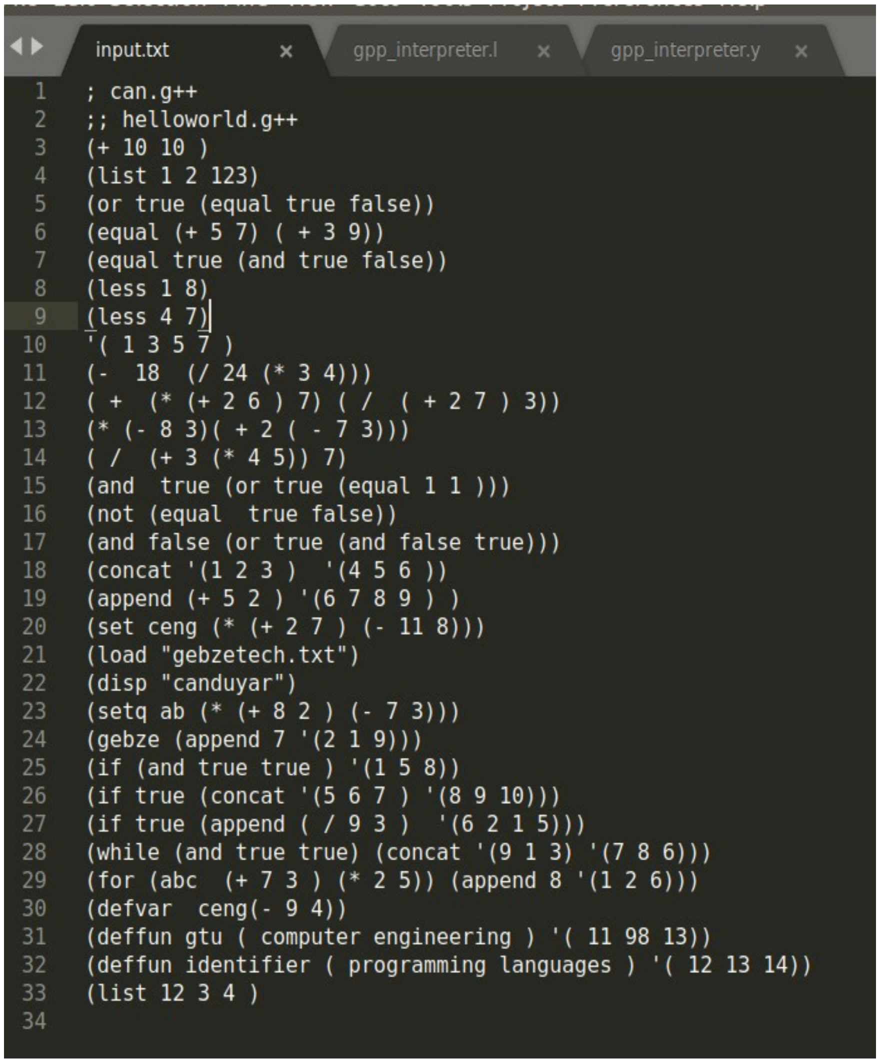
```
VALUES:
VALUES IntegerValue { A[ind]=$2; ind=ind+1; }
| IntegerValue  { A[ind]=$1 ; ind=ind+1; };

EXPI:
OP_OP OP_PLUS EXPI EXPI OP_CP {$$=$3+$4; checkPrint=1; number=1; }
| OP_OP OP_DIV EXPI EXPI OP_CP {$$=$3/$4; checkPrint=1; number=1;}
| OP_OP OP_MULT EXPI EXPI OP_CP {$$=$3*$4; checkPrint=1; number=1;}
| OP_OP KW_DBLMULT EXPI EXPI OP_CP {$$ = pow($3,$4); checkPrint=1; number=1;}
| OP_OP OP_MINUS EXPI EXPI OP_CP {$$=$3-$4; checkPrint=1; number=1;}
| OP_OP Id EXPLISTI OP_CP { $$= $3; condition=1; checkPrint=1;  }
| OP_OP KW_SET Id EXPI OP_CP { $$ = $4; checkPrint=0;printf("SYNTAX OK.\n");printf("Result: %d\n",$$);set_control = 1;}
| OP_OP KW_DEFFUN Id IDLIST EXPLISTI OP_CP
| OP_OP KW_SETQ Id EXPI OP_CP {checkPrint=1; number=1; condition=0; $$=$4;} /////ADDITIONAL FEATURE FOR ASSIGNMENT
| OP_OP KW_LOAD OP_DOUBLEQUOTE Id OP_DOUBLEQUOTE OP_CP
| Id
| OP_OP KW_LOAD OP_OC Id OP_CC OP_CP
| OP_OP KW_DISP  OP_DOUBLEQUOTE Id OP_DOUBLEQUOTE  OP_CP
| IntegerValue  {$$=$1;  }
| COMMENT
| OP_OP KW_IF EXPB EXPLISTI OP_CP {//JUST ONE CASE FOR "IF" IMPLEMENTATION -> YOU ALSO EXPLAINED IT ON MOODLE PAGE...
    checkPrint=1;
    if($3==1){
         condition=1;
    } else{
        condition=0;
    }
}
```

TESTS

Reading from file : (./gpp_interpreter.out input.txt).

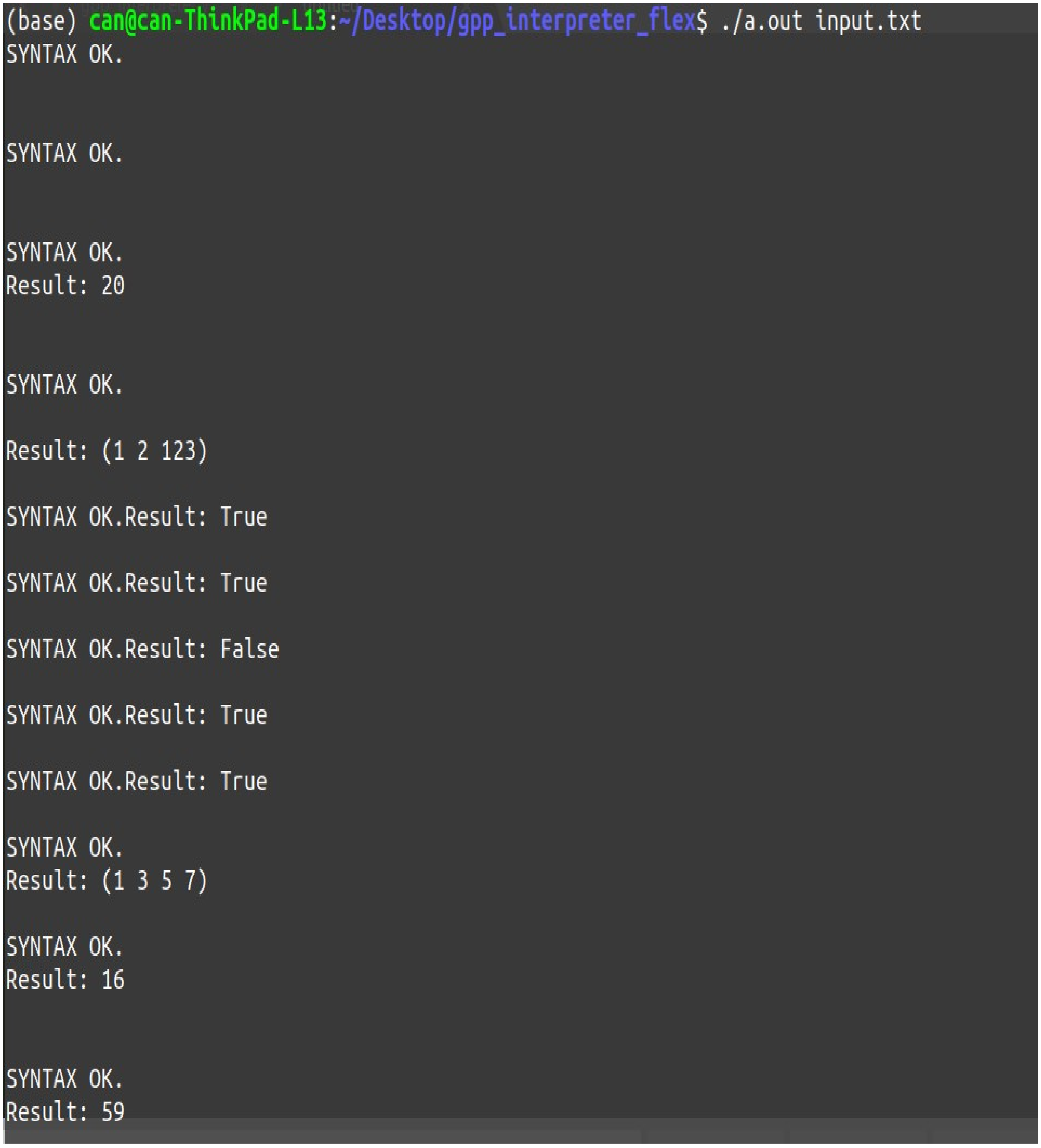
input.txt:

```
; can.g++
;; helloworld.g++
(+ 10 10 )
(list 1 2 123)
(or true (equal true false))
(equal (+ 5 7) ( + 3 9))
(equal true (and true false))
(less 1 8)
(less 4 7)
'( 1 3 5 7 )
(-  18  (/ 24 (* 3 4)))
( +  (* (+ 2 6 ) 7) ( /  ( + 2 7 ) 3))
(* (- 8 3)( + 2 ( - 7 3)))
( /  (+ 3 (* 4 5)) 7)
(and  true (or true (equal 1 1 )))
(not (equal  true false))
(and false (or true (and false true)))
(concat '(1 2 3 )  '(4 5 6 ))
(append (+ 5 2 ) '(6 7 8 9 ) )
(set ceng (* (+ 2 7 ) (- 11 8)))
(load "gebzetech.txt")
(disp "canduyar")
(setq ab (* (+ 8 2 ) (- 7 3)))
(gebze (append 7 '(2 1 9)))
(if (and true true ) '(1 5 8))
(if true (concat '(5 6 7 ) '(8 9 10)))
(if true (append ( / 9 3 )  '(6 2 1 5)))
(while (and true true) (concat '(9 1 3) '(7 8 6)))
(for (abc  (+ 7 3 ) (* 2 5)) (append 8 '(1 2 6)))
(defvar  ceng(- 9 4))
(deffun gtu ( computer engineering ) '( 11 98 13))
(deffun identifier ( programming languages ) '( 12 13 14))
(list 12 3 4 )
```

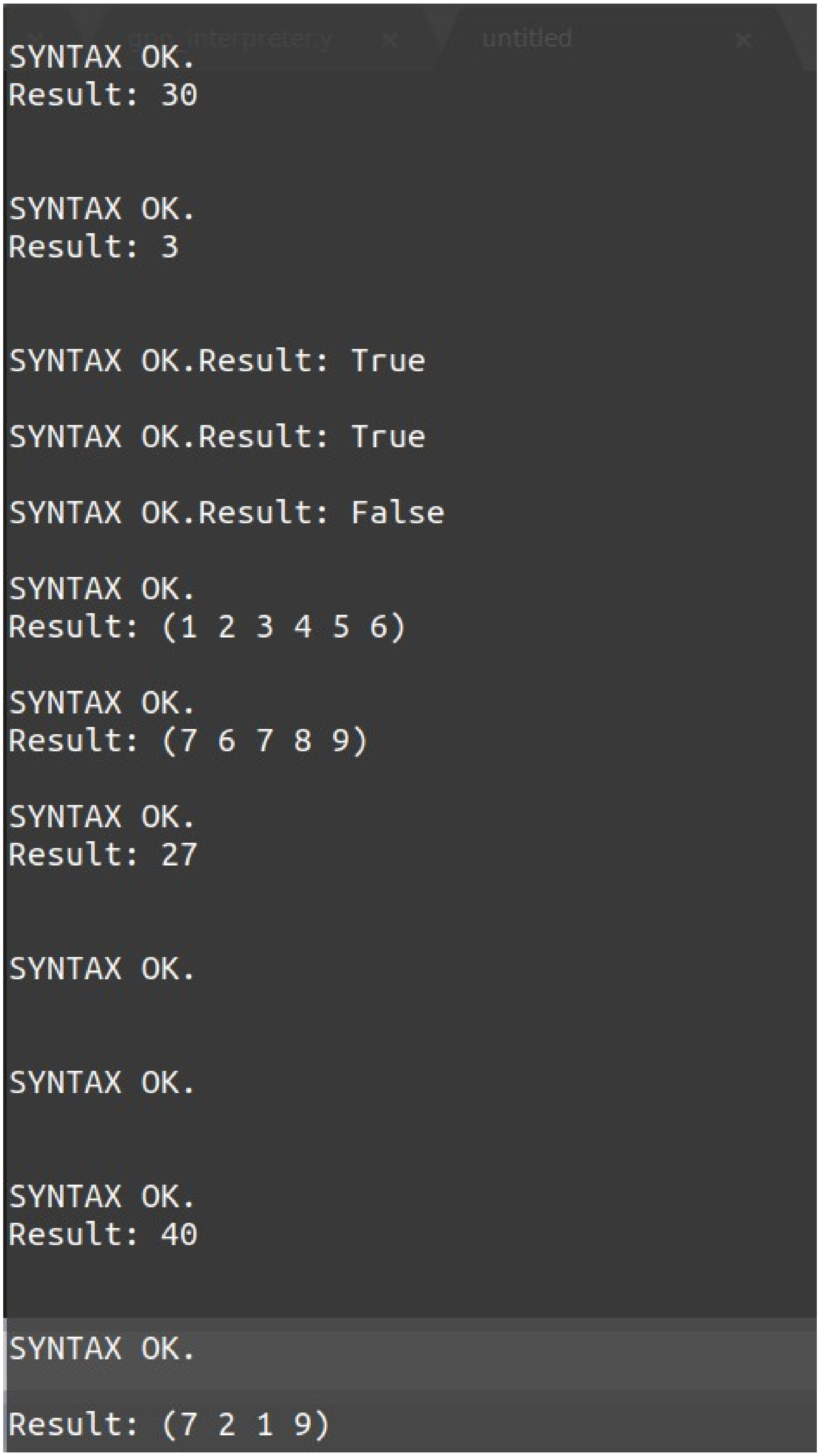


```
; can.g++
;; helloworld.g++
(+ 10 10 )
(list 1 2 123)
(or true (equal true false))
(equal (+ 5 7) ( + 3 9))
(equal true (and true false))
(less 1 8)
(less 4 7)
'( 1 3 5 7 )
(-  18  (/ 24 (* 3 4)))
( +  (* (+ 2 6 ) 7) ( /  ( + 2 7 ) 3))
(* (- 8 3)( + 2 ( - 7 3)))
( /  (+ 3 (* 4 5)) 7)
(and  true (or true (equal 1 1 )))
(not (equal  true false))
(and false (or true (and false true)))
(concat '(1 2 3 )  '(4 5 6 ))
(append (+ 5 2 ) '(6 7 8 9 ) )
(set ceng (* (+ 2 7 ) (- 11 8)))
(load "gebzetech.txt")
(disp "canduyar")
(setq ab (* (+ 8 2 ) (- 7 3)))
(gebze (append 7 '(2 1 9)))
(if (and true true ) '(1 5 8))
(if true (concat '(5 6 7 ) '(8 9 10)))
(if true (append ( / 9 3 )  '(6 2 1 5)))
(while (and true true) (concat '(9 1 3) '(7 8 6)))
(for (abc  (+ 7 3 ) (* 2 5)) (append 8 '(1 2 6)))
(defvar  ceng(- 9 4))
(deffun gtu ( computer engineering ) '( 11 98 13))
(deffun identifier ( programming languages ) '( 12 13 14))
(list 12 3 4 )
```

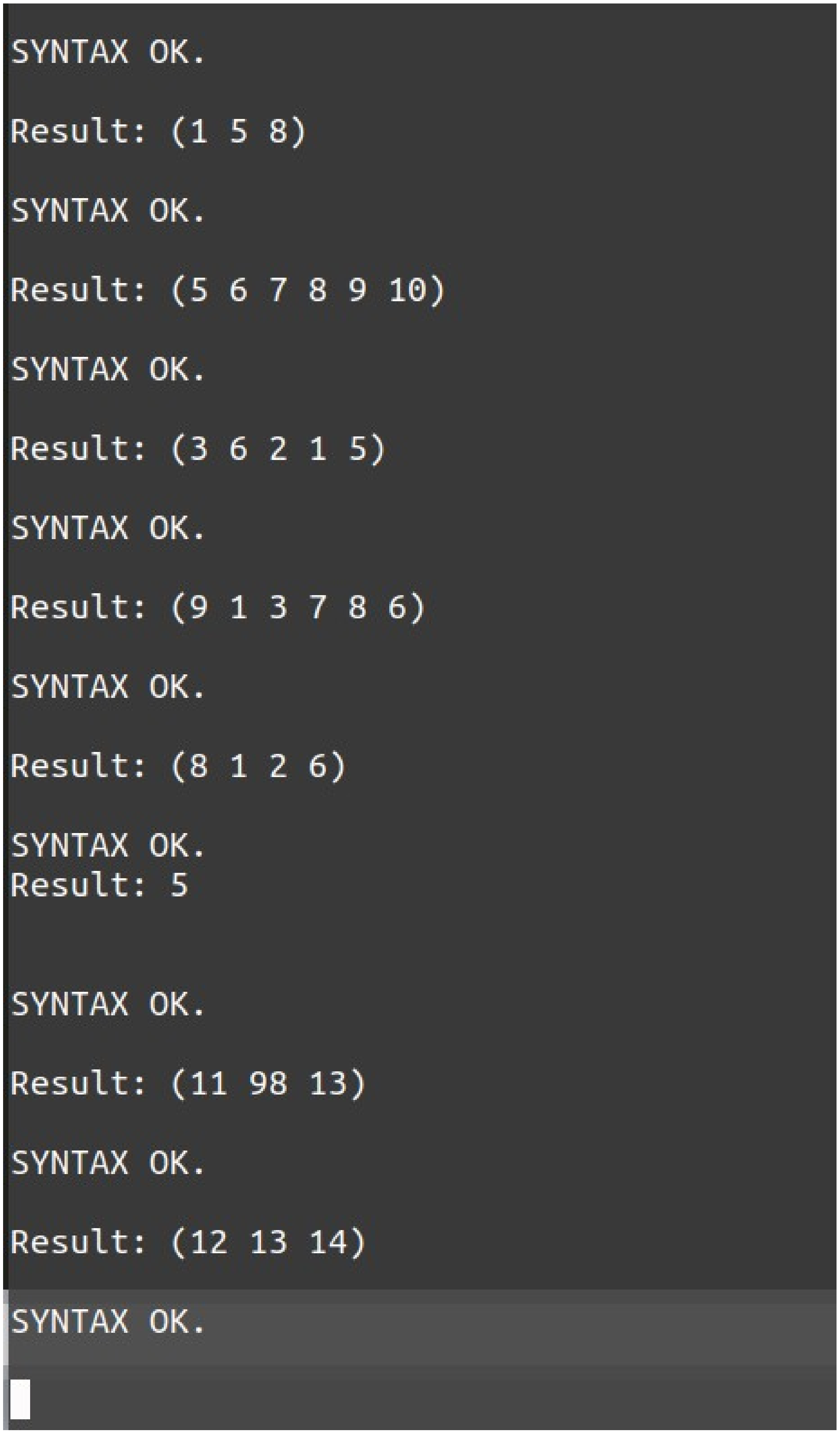
Results on terminal(results are according to the order above):



```
(base) can@can-ThinkPad-L13:~/Desktop/gpp_interpreter_flex$ ./a.out input.txt
SYNTAX OK.



SYNTAX OK.



SYNTAX OK.
Result: 20



SYNTAX OK.

Result: (1 2 123)

SYNTAX OK.Result: True

SYNTAX OK.Result: True

SYNTAX OK.Result: False

SYNTAX OK.Result: True

SYNTAX OK.Result: True

SYNTAX OK.
Result: (1 3 5 7)

SYNTAX OK.
Result: 16



SYNTAX OK.
Result: 59
```



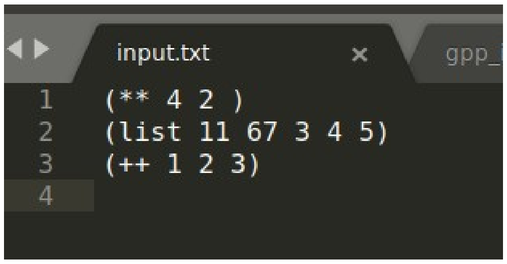
```
SYNTAX OK.
Result: 30


SYNTAX OK.
Result: 3


SYNTAX OK.Result: True

SYNTAX OK.Result: True

SYNTAX OK.Result: False

SYNTAX OK.
Result: (1 2 3 4 5 6)

SYNTAX OK.
Result: (7 6 7 8 9)

SYNTAX OK.
Result: 27


SYNTAX OK.


SYNTAX OK.


SYNTAX OK.
Result: 40


SYNTAX OK.

Result: (7 2 1 9)
```



```
SYNTAX OK.

Result: (1 5 8)

SYNTAX OK.

Result: (5 6 7 8 9 10)

SYNTAX OK.

Result: (3 6 2 1 5)

SYNTAX OK.

Result: (9 1 3 7 8 6)

SYNTAX OK.

Result: (8 1 2 6)

SYNTAX OK.
Result: 5


SYNTAX OK.

Result: (11 98 13)

SYNTAX OK.

Result: (12 13 14)

SYNTAX OK.
```

TEST-2 (with syntax error)

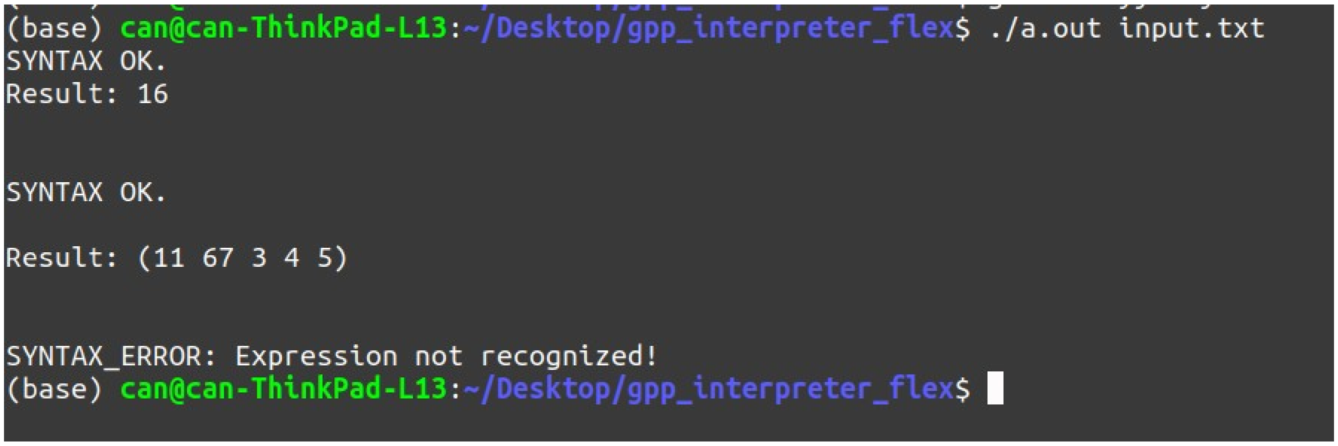
input.txt:



input.txt ×   gpp_

```
(** 4 2 )
(list 11 67 3 4 5)
(++ 1 2 3)
```

Results on terminal:

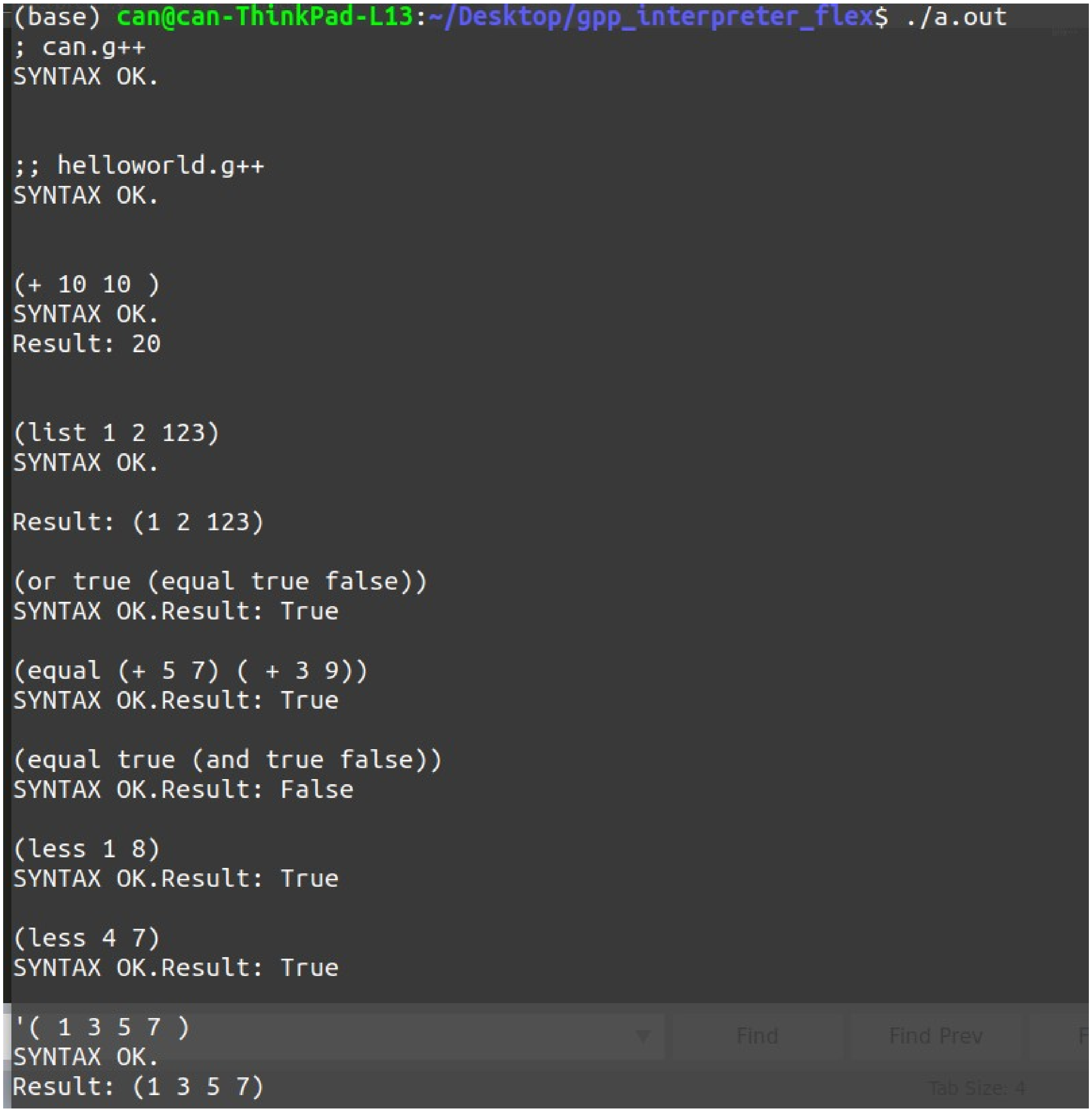


```
(base) can@can-ThinkPad-L13:~/Desktop/gpp_interpreter_flex$ ./a.out input.txt
SYNTAX OK.
Result: 16


SYNTAX OK.

Result: (11 67 3 4 5)


SYNTAX_ERROR: Expression not recognized!
(base) can@can-ThinkPad-L13:~/Desktop/gpp_interpreter_flex$
```

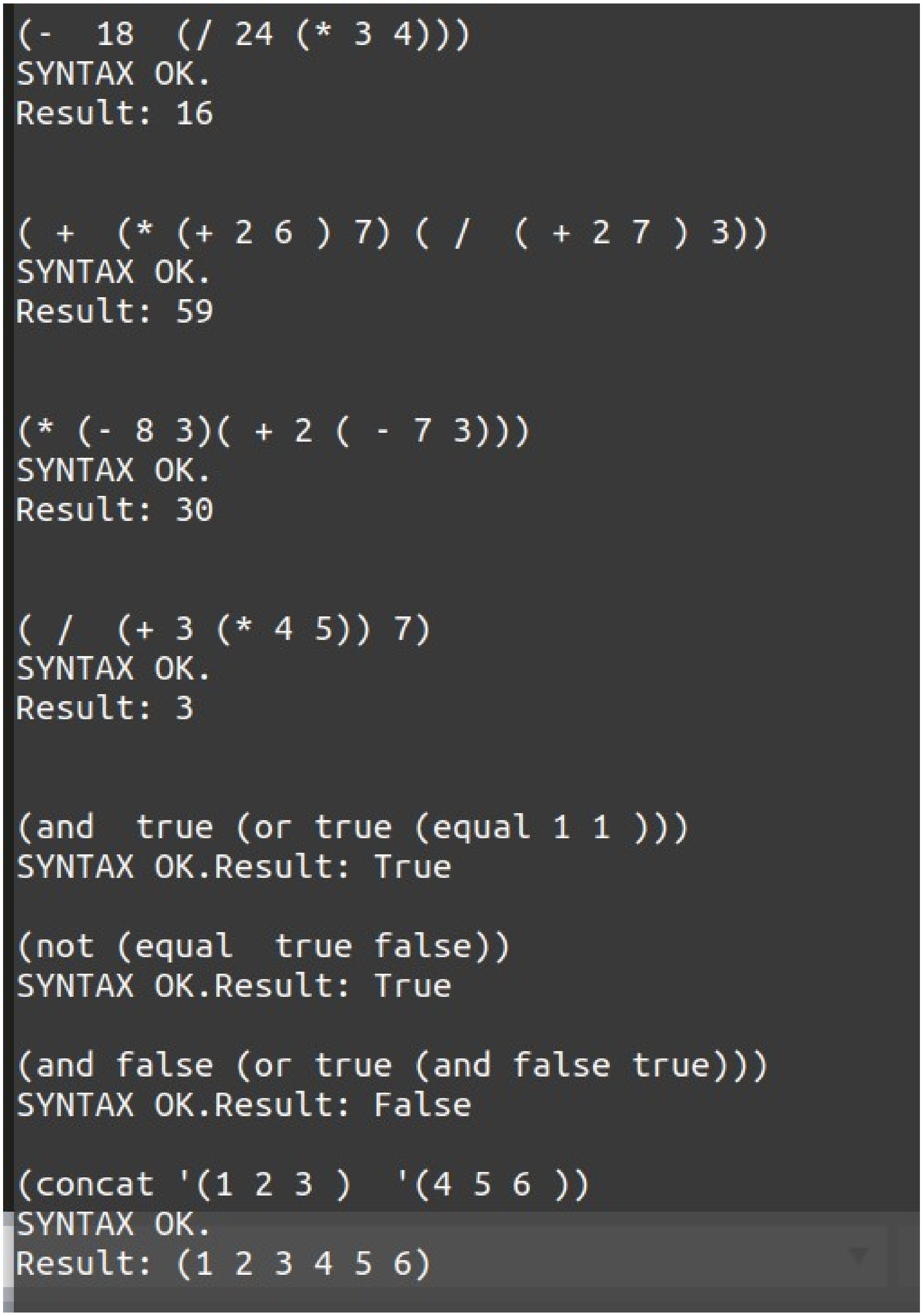
Running as interpreter directly: (without filename → ./gpp_interpreter.out)

* It takes inputs from terminal directly and works as an interpreter.

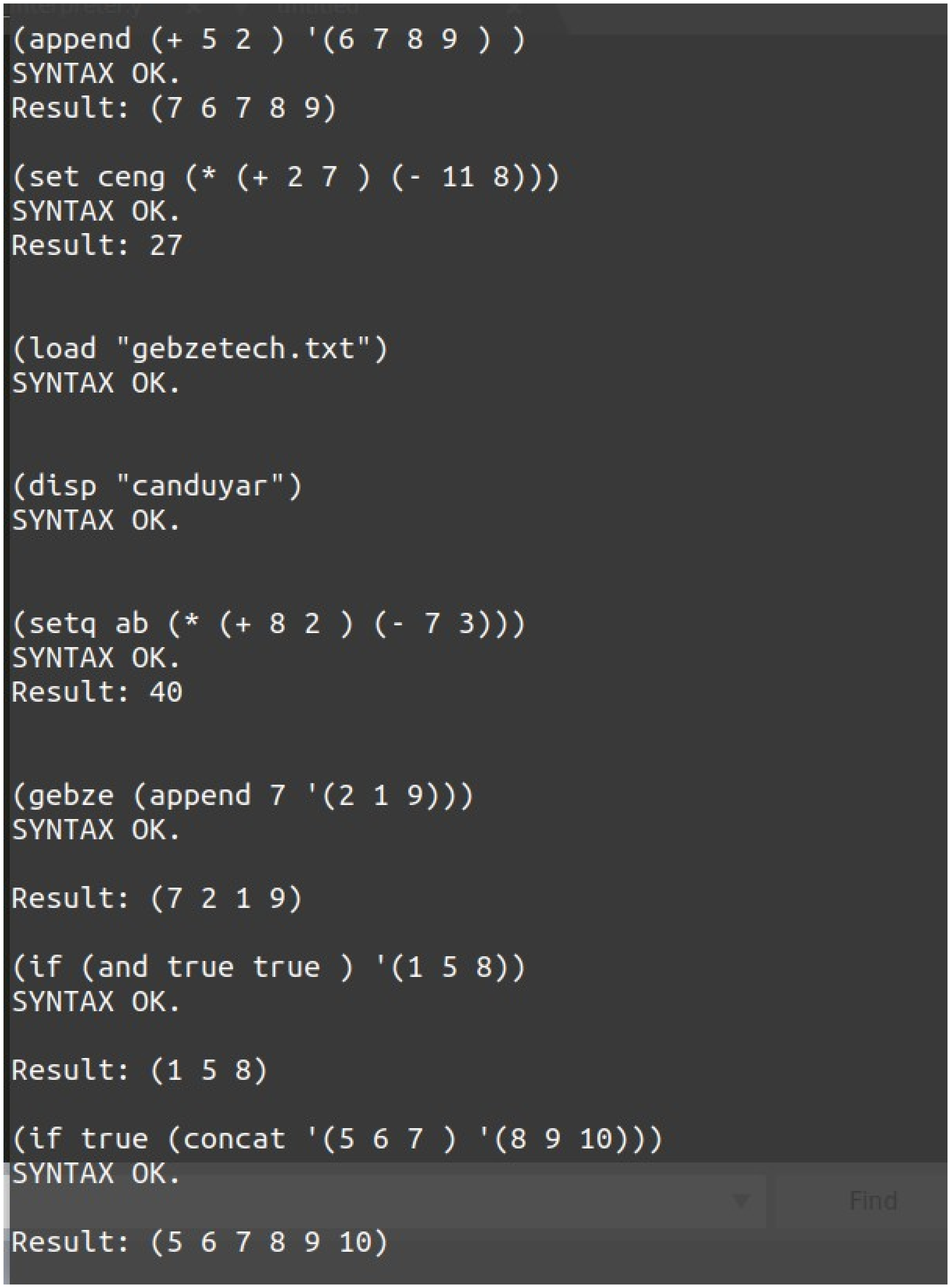
< TESTS >



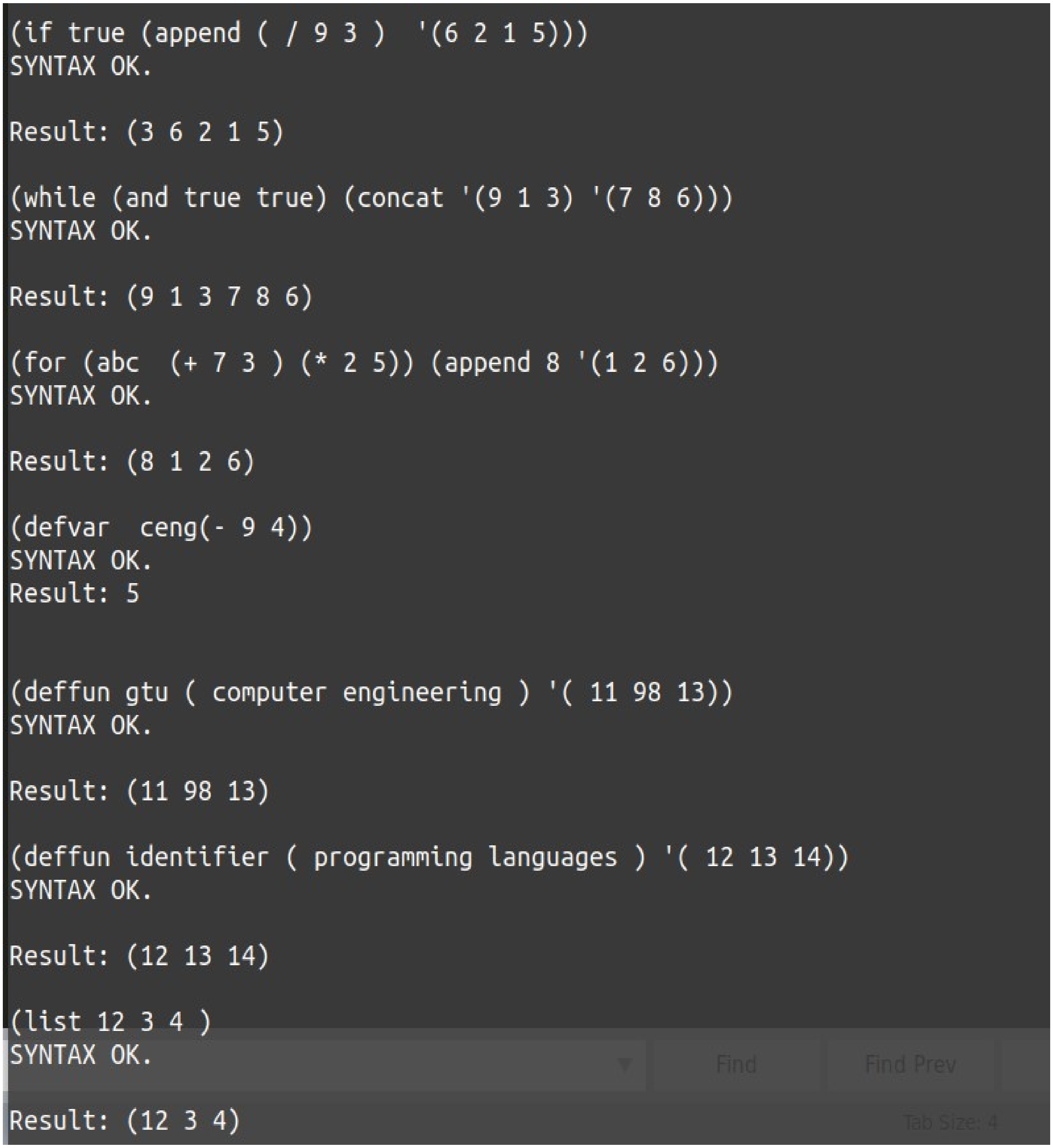
```
(base) can@can-ThinkPad-L13:~/Desktop/gpp_interpreter_flex$ ./a.out
; can.g++
SYNTAX OK.


;; helloworld.g++
SYNTAX OK.


(+ 10 10 )
SYNTAX OK.
Result: 20


(list 1 2 123)
SYNTAX OK.

Result: (1 2 123)

(or true (equal true false))
SYNTAX OK.Result: True

(equal (+ 5 7) ( + 3 9))
SYNTAX OK.Result: True

(equal true (and true false))
SYNTAX OK.Result: False

(less 1 8)
SYNTAX OK.Result: True

(less 4 7)
SYNTAX OK.Result: True

'( 1 3 5 7 )
SYNTAX OK.
Result: (1 3 5 7)
```

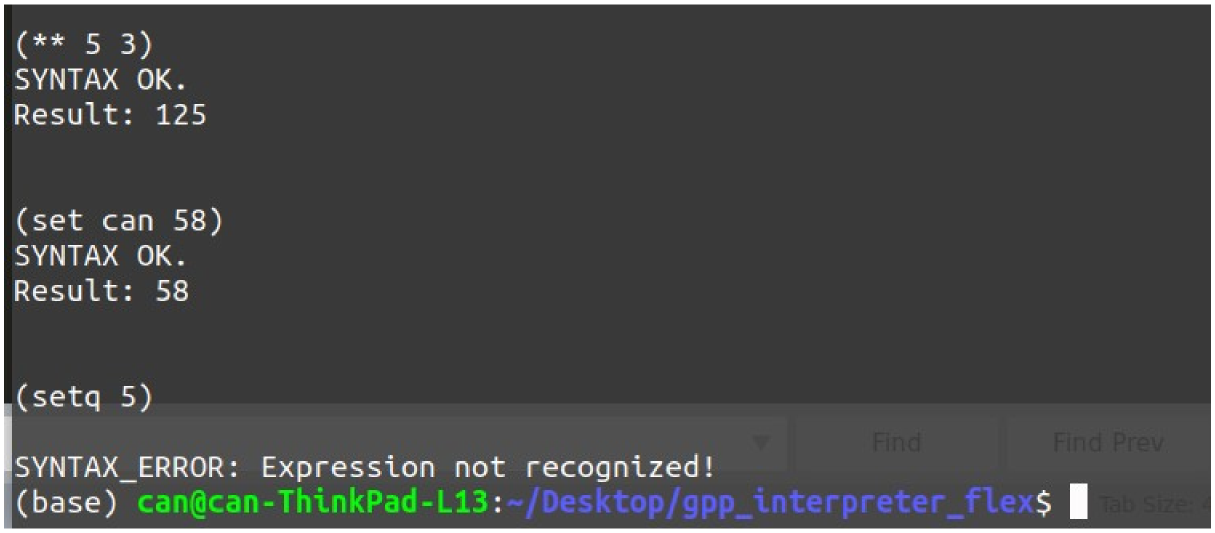


```
(-  18  (/ 24 (* 3 4)))
SYNTAX OK.
Result: 16


( +  (* (+ 2 6 ) 7) ( /  ( + 2 7 ) 3))
SYNTAX OK.
Result: 59


(* (- 8 3)( + 2 ( - 7 3)))
SYNTAX OK.
Result: 30


( /  (+ 3 (* 4 5)) 7)
SYNTAX OK.
Result: 3


(and  true (or true (equal 1 1 )))
SYNTAX OK.Result: True

(not (equal  true false))
SYNTAX OK.Result: True

(and false (or true (and false true)))
SYNTAX OK.Result: False

(concat '(1 2 3 )  '(4 5 6 ))
SYNTAX OK.
Result: (1 2 3 4 5 6)
```



```
(append (+ 5 2 ) '(6 7 8 9 ) )
SYNTAX OK.
Result: (7 6 7 8 9)

(set ceng (* (+ 2 7 ) (- 11 8)))
SYNTAX OK.
Result: 27


(load "gebzetech.txt")
SYNTAX OK.


(disp "canduyar")
SYNTAX OK.


(setq ab (* (+ 8 2 ) (- 7 3)))
SYNTAX OK.
Result: 40


(gebze (append 7 '(2 1 9)))
SYNTAX OK.

Result: (7 2 1 9)

(if (and true true ) '(1 5 8))
SYNTAX OK.

Result: (1 5 8)

(if true (concat '(5 6 7 ) '(8 9 10)))
SYNTAX OK.

Result: (5 6 7 8 9 10)
```


```
(if true (append ( / 9 3 )  '(6 2 1 5)))
SYNTAX OK.

Result: (3 6 2 1 5)

(while (and true true) (concat '(9 1 3) '(7 8 6)))
SYNTAX OK.

Result: (9 1 3 7 8 6)

(for (abc  (+ 7 3 ) (* 2 5)) (append 8 '(1 2 6)))
SYNTAX OK.

Result: (8 1 2 6)

(defvar  ceng(- 9 4))
SYNTAX OK.
Result: 5


(deffun gtu ( computer engineering ) '( 11 98 13))
SYNTAX OK.

Result: (11 98 13)

(deffun identifier ( programming languages ) '( 12 13 14))
SYNTAX OK.

Result: (12 13 14)

(list 12 3 4 )
SYNTAX OK.

Result: (12 3 4)
```



```
(** 5 3)
SYNTAX OK.
Result: 125


(set can 58)
SYNTAX OK.
Result: 58


(setq 5)

SYNTAX_ERROR: Expression not recognized!
(base) can@can-ThinkPad-L13:~/Desktop/gpp_interpreter_flex$
```

G++ Language Interpreter in Lisp(All parts work correctly):

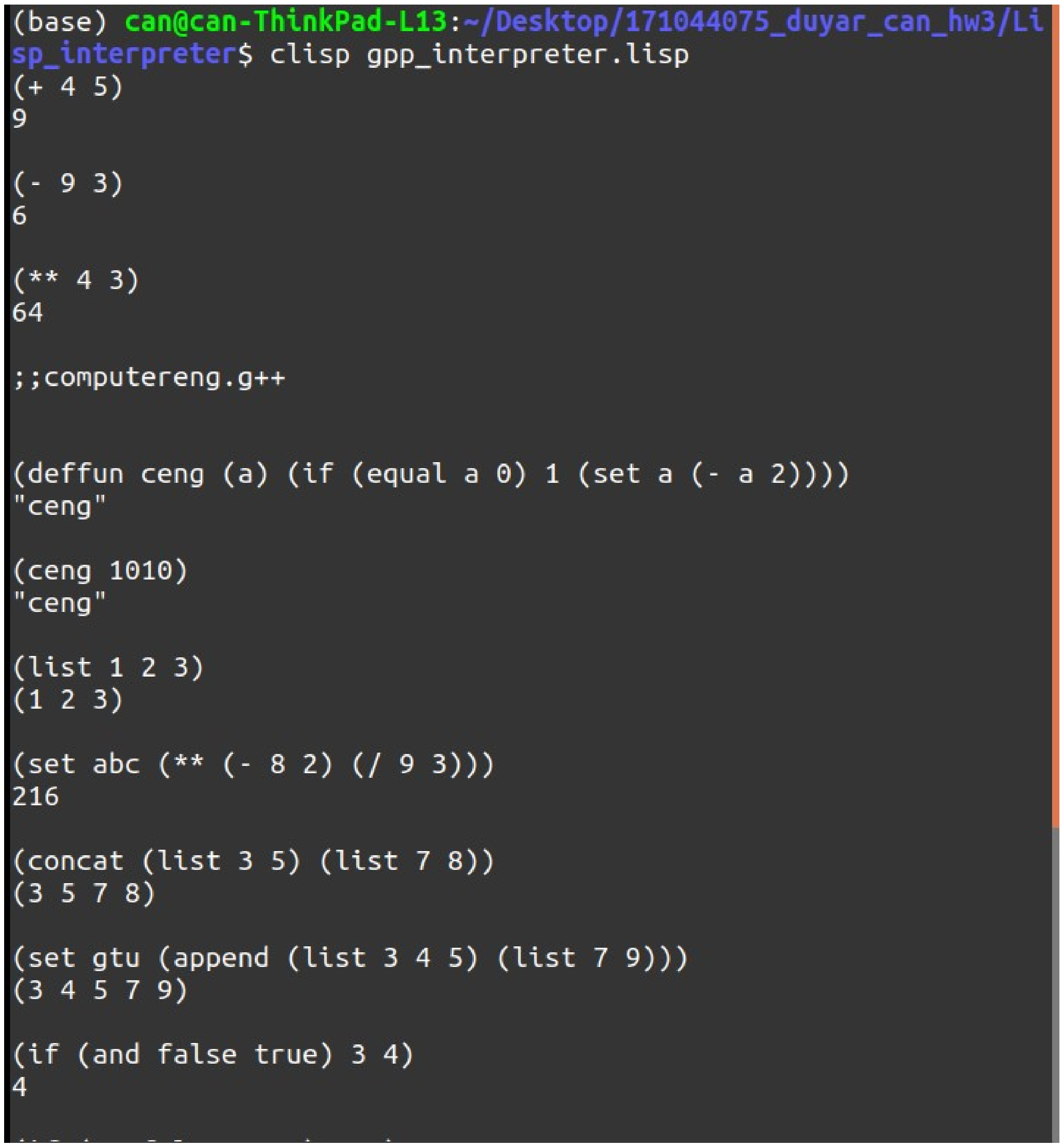
I implemented my interpreter in common lisp. In this interpreter design, I have a function called “*gppinterpreter*”*. This function starts my interpreter according to number of input. If it has zero input then it means that user didn’t enter any filename as a terminal argument so it continuous as normal interpreter.If it has one input then it means that user entered a filename as a terminal argument then my gppinterpreter function starts with file and reads from it.*

*I Implemented function definition and the body (deffun)
**I also implemented assignment operation with “set” in this part according to moodle-page instructions.*

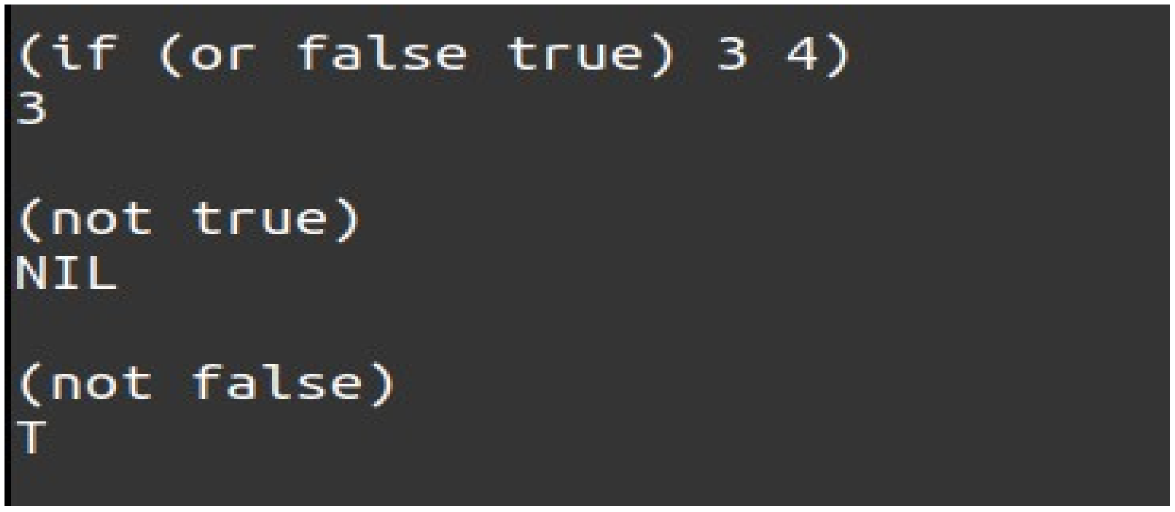
→ *I didn’t write “Syntax Ok.” if its true then it doesn’t return(if it returns a value then prints it)*
→ *if syntax is a problem for given input then it prints “SYNTAX_ERROR Expression not recognized”*

<TESTS>

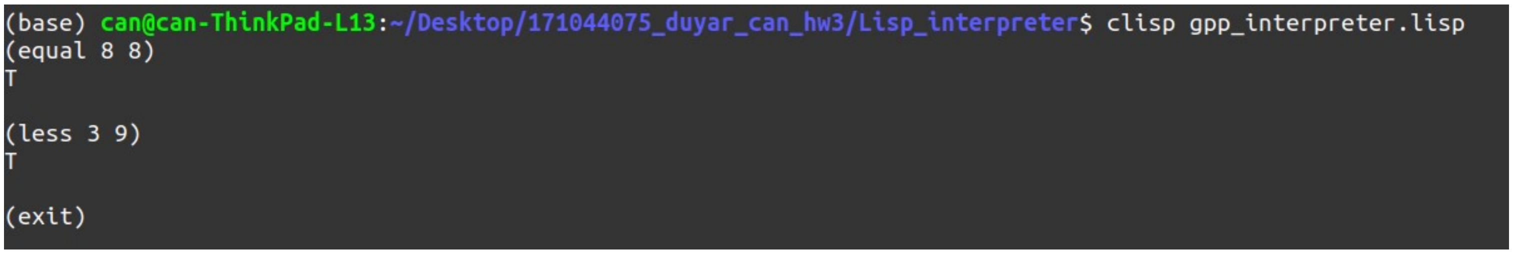
Starting as normal interpreter:without filename → clisp gpp_interpreter.lisp)



```
(base) can@can-ThinkPad-L13:~/Desktop/171044075_duyar_can_hw3/Li
sp_interpreter$ clisp gpp_interpreter.lisp
(+ 4 5)
9

(- 9 3)
6

(** 4 3)
64

;;computereng.g++


(deffun ceng (a) (if (equal a 0) 1 (set a (- a 2))))
"ceng"

(ceng 1010)
"ceng"

(list 1 2 3)
(1 2 3)

(set abc (** (- 8 2) (/ 9 3)))
216

(concat (list 3 5) (list 7 8))
(3 5 7 8)

(set gtu (append (list 3 4 5) (list 7 9)))
(3 4 5 7 9)

(if (and false true) 3 4)
4
```



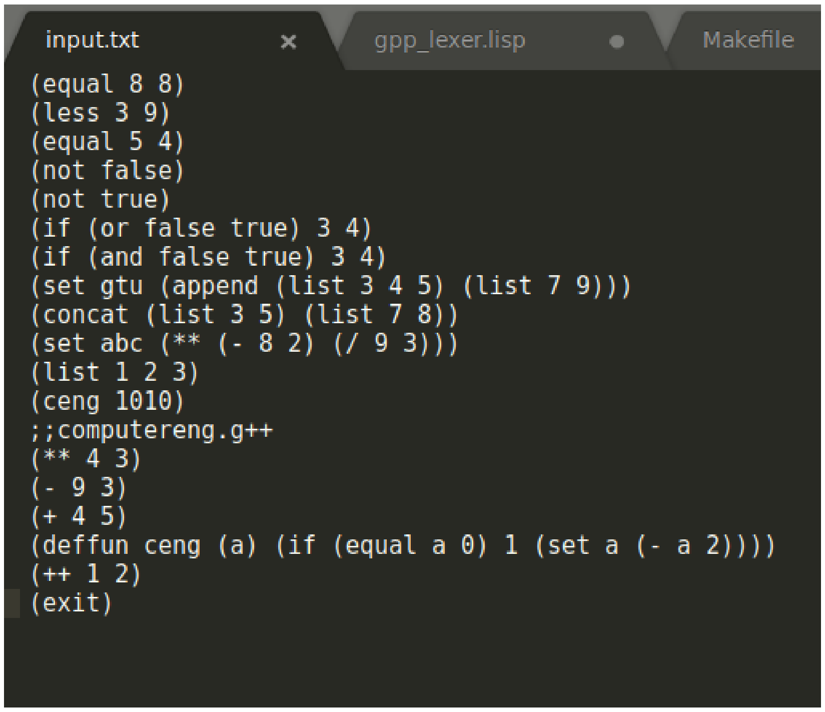
```
(if (or false true) 3 4)
3

(not true)
NIL

(not false)
T
```



```
(base) can@can-ThinkPad-L13:~/Desktop/171044075_duyar_can_hw3/Lisp_interpreter$ clisp gpp_interpreter.lisp
(equal 8 8)
T

(less 3 9)
T

(exit)
```

Interpreter with a filename.txt: → *clisp gpp_interpreter.lisp input.txt*

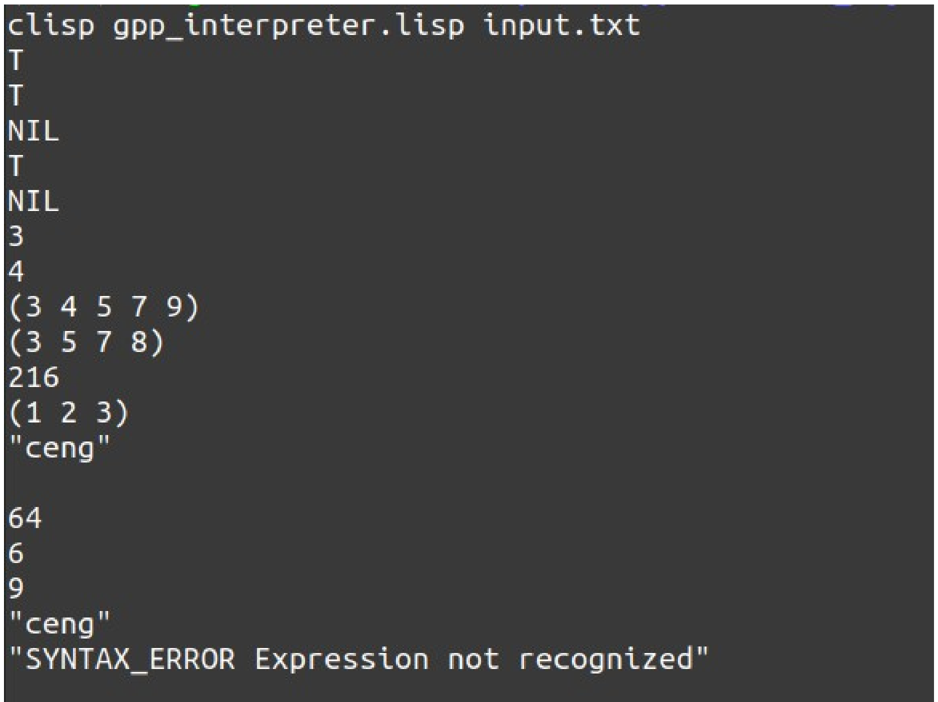
input.txt



input.txt × gpp_lexer.lisp ● Makefile

```
(equal 8 8)
(less 3 9)
(equal 5 4)
(not false)
(not true)
(if (or false true) 3 4)
(if (and false true) 3 4)
(set gtu (append (list 3 4 5) (list 7 9)))
(concat (list 3 5) (list 7 8))
(set abc (** (- 8 2) (/ 9 3)))
(list 1 2 3)
(ceng 1010)
;;computereng.g++
(** 4 3)
(- 9 3)
(+ 4 5)
(deffun ceng (a) (if (equal a 0) 1 (set a (- a 2))))
(++ 1 2)
(exit)
```

terminal:



```
clisp gpp_interpreter.lisp input.txt
T
T
NIL
T
NIL
3
4
(3 4 5 7 9)
(3 5 7 8)
216
(1 2 3)
"ceng"

64
6
9
"ceng"
"SYNTAX_ERROR Expression not recognized"
```