GTU Department of Computer Engineering

CSE341 - Fall 2020

Homework 3(Yacc and Lisp Interpreter) Report

Akif Safa Angi

200104004079

For yacc:

Execute program writing “make” or “make input” to input test file

For lisp:

Execute program writing clisp gpp_interpreter.lisp

After starting program, to read from file enter file name or enter start to get input from user

Part 1

Working syntax;

+, -, /, *, (), ;;, def

Operator Tests

```
Type (exit) for exit
Enter your input
(+ 4b2 8b4)
Result = 32b8
(- 9b3 8b5)
Result = 21b15
(/ 11b4 2b7)
Result = 77b8
(* 3b8 9b2)
Result = 27b16
(+ 4b2 (* 2b5 4b6))
Result = 136b60
(/ (* 2b3 5b7) (+ 4b2 4b6))
Result = 120b672
(exit)
```

Defining two, one and non-parameter Functions

```
a@DESKTOP-5F6OAM7:/mnt/c/Users/akif_/Desktop/lisp/hw3/part 1$ make
flex gpp_lexer.l
yacc -d gpp_interpreter.y
gcc -g lex.yy.c y.tab.c gpp_helper.c -ll -o a.out
./a.out
Type (exit) for exit
Enter your input
(def twopar x y (+ x y))
(twopar 2b4 4b6)
Result = 28b24
(def onepar z (* z 3b4))
(onepar 6b8)
Result = 18b32
(def nonpar (- 8b4 6b5))
(nonpar)
Result = 16b20
(exit)
```

Using function inside of an another function

```
a@DESKTOP-5F6OAM7: /mnt,
a@DESKTOP-5F6OAM7:/mnt/c/Users/akif_/Desktop/lisp/hw3/part 1$ make
flex gpp_lexer.l
yacc -d gpp_interpreter.y
gcc -g lex.yy.c y.tab.c gpp_helper.c -ll -o a.out
./a.out
Type (exit) for exit
Enter your input
(def sum x y (+ x y))
(def sum2 x (* 3b4 x))
(sum (sum2 6b7) 2b3)
Result = 110b84
```

Nested one parameter function

```
Enter your input
(def div x (/ 4b5 x))
(div (div 8b6))
Result = 160b120
```

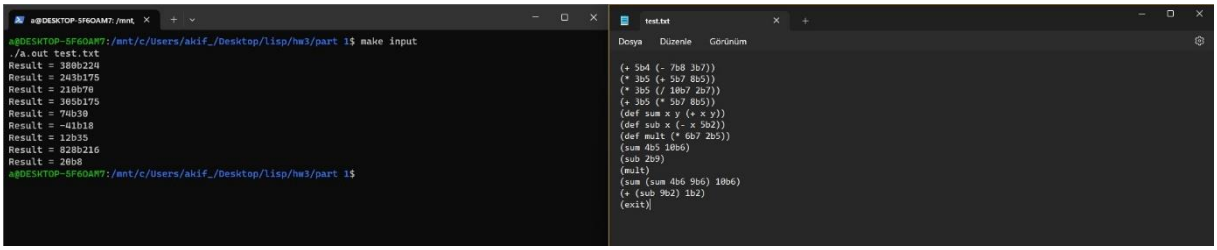
Nested two parameter function

```
Enter your input
(def a x y (+ x y))
(a 2b3 (a 4b6 6b8))
Result = 300b144
(a (a 2b2 4b2) 8b4)
Result = 80b16
```

Nested expression for function

```
a@DESKTOP-5F6OAM7: /mnt,
a@DESKTOP-5F6OAM7:/mnt/c/Users/akif_/Desktop/lisp/hw3/part 1$ make
flex gpp_lexer.l
yacc -d gpp_interpreter.y
gcc -g lex.yy.c y.tab.c gpp_helper.c -ll -o a.out
./a.out
Type (exit) for exit
Enter your input
(def nest1 y z (+ z (* 2b6 y)))
(def nest2 x (* 3b5 (- 4b6 x)))
(def nest3 (/ 2b2 (+ 3b4 4b8)))
(nest1 3b9 9b8)
Result = 534b432
(nest2 3b4)
Result = -6b120
(nest3)
Result = 64b80
(exit)
```

Test File



Part 2

Working syntax;

+, -, /, *, (), def, ;;

Execute program writing “clisp gpp_interpreter.lisp”

Operator Tests

```
a@DESKTOP-5F6OAM7:/mnt/c/Users/akif_/Desktop/lisp/hw3/part 2$ clisp gpp_interpreter.lisp
If you wanna read from a file enter filename. Else enter start.
start

>>> (+ 4b5 6b7)
Result: 58b35

>>> (- 12b4 3b2)
Result: 3b2

>>> (/ 7b5 9b4)
Result: 28b45

>>> (* 3b9 12b6)
Result: 2b3

>>> (+ 2b1 (- 12b10 3b6))
Result: 27b10

>>> (* (+ 3b2 6b3) (/ 8b2 9b2))
Result: 28b9

>>> (exit)
```

Defining and calling non-parameter function

```
If you wanna read from a file enter filename. Else enter start.
start

>>> (def non (+ 2b2 5b5))
Function Created

>>> (non)
Result: 2b1
```

Defining and calling one parameter function

```
If you wanna read from a file enter filename. Else enter start.
start

>>> (def mul x (* 5b6 x))
Function Created

>>> (mul 2b3)
Result: 5b9

>>> (mul (mul 11b2))
Result: 275b72
```

Defining and calling two parameters function

```
If you wanna read from a file enter filename. Else enter start.
start

>>> (def a x y(+ x y))
Function Created

>>> (a 4b5 5b5)
Result: 9b5

>>> (a 4b5 (a 5b5 6b5))
Result: 3b1

>>> (a (a 9b6 7b8) 2b4)
Result: 23b8
```

Defining and calling nested functions

```
>>> (def tnest x z (+ x (* 4b2 z)))
Function Created

>>> (def onest z (/ z (+ 2b1 6b7)))
Function Created

>>> (def nest (* 2b2 (+ 5b4 2b3)))
Function Created

>>> (tnest 3b6 12b8)
Result: 7b2

>>> (onest 10b3)
Result: 7b6

>>> (nest)
Result: 23b12

>>> (exit)
```

Call one function in another

```
>>> (def a x y (+ x y))
Function Created

>>> (def b x (* 3b4 x))
Function Created

>>> (a (b 6b7) 2b3)
Result: 55b42
```

Test File