Assignemnt 2: Lexical Analyzer and Parser for Boolean Expressions

System Programming 2023

Assigned: 19th October 2023 Due: 2nd November 2023

1 Task

Write a program that reads from terminal user-entered boolean expression. Boolean expressions have boolean operators (see table 1) applied to logical values (0 or 1).
The logical values can be supplied by:
(A) directly in the expression, ex: 0+1, or
(B) can be assigned to a variable. The variables are represented by any letter followed by a number, ex: x1, t2, s26 - are all variables; values can be assigned to these variables using equality operator, ex: x1 = 0.

The following questions need to be implemented as per the input:

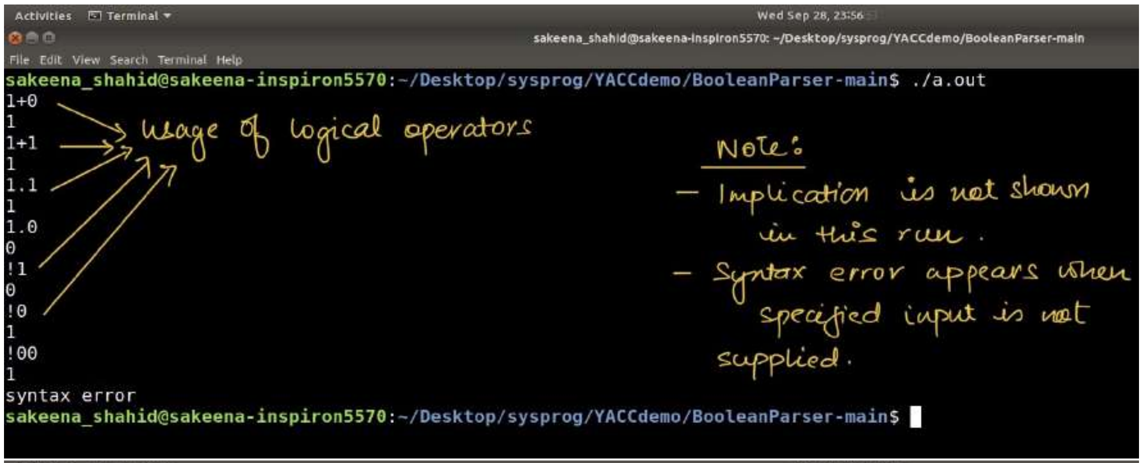
1. Enter expression from the terminal, and upon hitting enter, the value of the boolean expression should be displayed.

2. Use bitwise operators (see table 2[1]) on variables[2].

[1]Other operators such as left and right shift are also possible but not mandatory for this assignment.

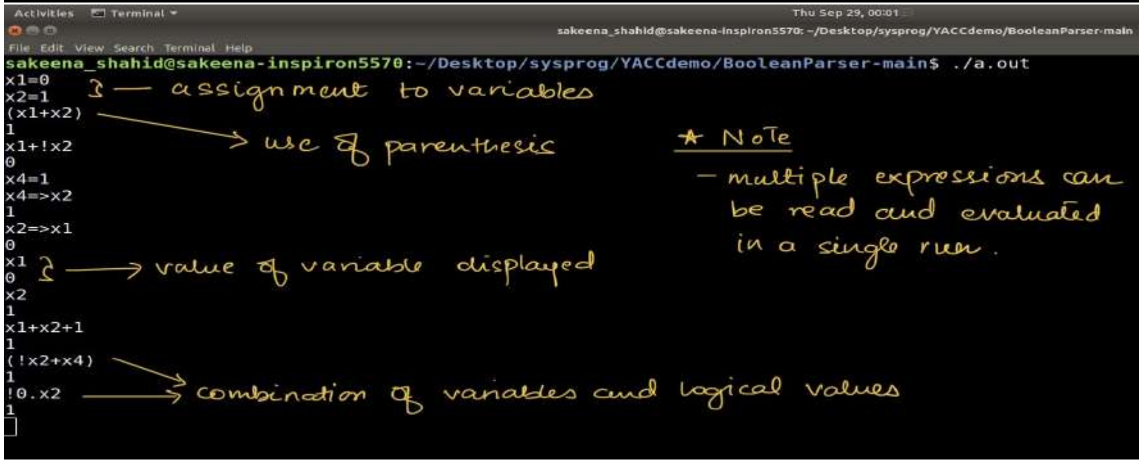
[2]Essentially we should be able to supply input in decimal.binary representation to these operators, howver, for the purposes of this assignment, applying them to 1 bit of 0 or 1 is okay. Youi can implement it for extra credit.

2 Sample Run


Activities Terminal
Wed Sep 28, 23:56
sakeena_shahid@sakeena-inspiron5570: ~/Desktop/sysprog/YACCdemo/BooleanParser-main
File Edit View Search Terminal Help
usage of logical operators

```
sakeena_shahid@sakeena-inspiron5570:~/Desktop/sysprog/YACCdemo/BooleanParser-main$ ./a.out
1+0
1
1+1
1
1.1
1
1.0
0
!1
0
!0
1
!00
1
syntax error
sakeena_shahid@sakeena-inspiron5570:~/Desktop/sysprog/YACCdemo/BooleanParser-main$
```

Note:
- Implication is not shown in this run.
- Syntax error appears when specified input is not supplied.


Activities Terminal
Thu Sep 29, 00:01
sakeena_shahid@sakeena-inspiron5570: ~/Desktop/sysprog/YACCdemo/BooleanParser-main
File Edit View Search Terminal Help
} — assignment to variables
use of parenthesis
} value of variable displayed
combination of variables and logical values

```
sakeena_shahid@sakeena-inspiron5570:~/Desktop/sysprog/YACCdemo/BooleanParser-main$ ./a.out
x1=0
x2=1
(x1+x2)
1
x1+!x2
0
x4=1
x4=>x2
1
x2=>x1
0
x1
0
x2
1
x1+x2+1
1
(!x2+x4)
1
!0.x2
1
```

★ Note
- multiple expressions can be read and evaluated in a single run.

3 Submission

In lab-class discussion of your work. All files (input, lex, yacc) to be placed at appropriate places for easy compilation.
Please work on this assignment individually. If you think your programming is weak, then the assignment can be completed in groups of 2 (marks will be given accordingly).

Operator	**Functionality**
+	logical OR
.	logical AND
!	logical NOT
=>	implication
(boolean expression)	parenthesis have their usual meaning

Table 1: Table of Boolean Operators

Operator	**Functionality**
&	Bitwise AND
|	Bitwise OR
^	Bitwise NOT

Table 2: Table of Bitwise Operators

Assignemnt 1: Lexical Analyzer for CEasy

System Programming August 2023

Assigned: October 05, 2023 **Due: October 12, 2023**

1 Task

Write a program that reads an input file, and constructs a list of tokens for that file. Your program is to be written in CEasy. The list of tokens accepted by CEasy is shown in Table 1.
The following questions also need to be implemented as per the input file:

1. Count the number of words, characters, blank spaces, and lines
2. Create another file where the lines from input file are numbered

2 Sample Input

```
void main()
{
        int sum =0;
        for(int i=0; i<10; i=i+1)
        {
                sum = sum + i + 10.43 + 34.E4 + 45.34E-4 + E43 + .23;
        }
}
```

3 Sample Output

```
Class : Lexeme
keyword : void
identifier : main
( : (
) : )
{ : {
keyword : int
identifier : sum
= : =
num : 0
; : ;
keyword : for
( : (
keyword : int
identifier : i
= : =
num : 0
; : ;
identifier : j
< : <
num : 10
; : ;
identifier : i
= : =
identifier : i
+ : +
num : 1
) : )
{ : {
identifier : sum
= : =
identifier : sum
+ : +
identifier : i
+ : +
num : 10.43
+ : +
num : 34.E4
+ : +
num: 45.34E-4
+ : +
identifier : E43
+ : +
Error : .
num : 23
; : ;
} : }
} : }
```

4 Submission

In lab-class discussion of your work. All files (input, lex) to be placed at appropriate places for easy compilation.
Please work on this assignment individually. If you think your programming is weak, then the assignment can be completed in groups of 2 (marks will be given

Token Type	**Lexical Specification**
keyword	One of the strings **while, if, else, return, break, continue, int, float, void**

identifier	Taken Id identifiers matches a letter followed by letters or digits or underscore
	`letter -> [A-Za-z]`
	`digit ->[0-9]`
	`id -> letter(letter | digit _ ) *`

num	Token num matches unsigned numbers
	`digits -> digit digit*`
	`optional-fraction -> (.digits)|` ϵ
	`optional-exponent -> (E(+|-|` ϵ `)digits)|` ϵ
	`num -> digits optional-fraction optional-exponent`

addop	`+,-`
mulop	`*,/`
relop	`<,>,>=,<=,==,!=`
and	`&&`
or	`||`
not	`!`
`)`	`(`
`)`	`)`
`{`	`{`
`}`	`}`
`[`	`[`
`]`	`]`

Table 1: Table of Lexical specifications for tokens

accordingly).