Assignment 1 – Programming In-class assignment 2024

Consider the following grammar for a simple calculator that can handle any number of integers.

```
statements ::= { statement '\n' } | statement STOP
statement ::= expression
expression ::= term   | expression '+' term   | expression '-' term
term ::= factor  | term '*' factor     | term '/' factor
factor ::= NUMBER  | '(' expression ')'
```

- upon arriving STOP, program should terminate.
- Tokens may be separated by any number of white spaces, tabs or new lines.
- Expressions are followed by + or – operator. Note that these are left associative and have lower precedence than multiplication and division.
- Term is followed by * or / operator and they are left associative.
- If there is a parse error, you need to report the error.
    - Your error messages must contain the line number where the error was found.

Submission Guidelines

Hand in your source code electronically (do not submit .o or executable code).

Each group uploads only ONE copy of the assignment

Make sure that this code compiles and runs correctly on linux. The makefile must give the executable code the name calc

Write a README file (text file, do not submit a .doc file) which contains

- Names, section numbers, and email addresses of group members
- Whether your code was tested on linux.
- How to execute your program.
- Briefly describe anything special about your submission that the instructor should take note of.

Place all your files under one directory with group name (p1-GroupNo for assignment 1, e.g. p1- G2, your corresponding group number can be found in shared google sheet).

Tar the contents of this directory using the command

tar –cvf [directory_name].tar [directory_name]

E.g. tar -cvf p1G2.tar p1-G2/

Academic Integrity

- We will use moss to detect to plagiarism in this assignment.
- Use the UGVLE to upload the tared file you created above.

Compile your code

flex –l calc.l

bison -dv calc.y

gcc -o calc calc.tab.c lex.yy.c –lfl

Execution (example):.

./calc

Example testcases

1. 1+3
   Result: 4

2. 5-3
   Result: 2

3. 4*2+5
   Result: 13

4. 2+4-5
   Result: 1

5. 8/2
   Result: 4

6. (2+3)*5
   Result: 25

7. (1 + (2 + 3)) * 4
   Result: 24

8. 10-2+3
   Result: 11

9. 5/0
   Error: Division by zero
   Result: 0

10. 2 + 3 * (4 - 1)
    Result: 11

11. 3 + 5 * 2 - 8 / 4
    Result: 11

12. ((2 + 3) * 4) / 2
    Result: 10

13. 42
    Result: 42

14. (3 * 2) + 5
    Result: 11

15. 5 + (3 * 2)
    Result: 11

16. 1 + 1
    Result: 2

17. 2 * 2
    Result: 4

18. 3 * (2 + (5 - 3) * 2)
    Result: 18

19. 1+(1)
    Result: 2

20. 5 + 3 STOP
    Result: 8
    Program terminates