**Goal:**

1. Learn how to write and use makefile: <http://www.delorie.com/djgpp/doc/ug/larger/makefiles.html>

2. Acquaint yourself with flex and bison.

***Specifications***

**You will extend calc.l and calc.y to parse and type check programs whose syntax is defined below.**

Prog 🡪 main() {Stmts}

Stmts 🡪 **ε** | Stmt; Stmts

Stmt 🡪 int Id | float Id | Id = E | **printvar** Id

E 🡪 Integer | Float | Id | E + E | E \* E

Integer 🡪 digit+

Float 🡪 Integer . Integer

**Stmts** is a sequence of statements. **Id** is an identifier, which is a sequence of one or more lower-case letters or digits. **Id** should start with a lower-case letters. For example, x, x1, xy are identifiers, but 1x and A are not. Each variable is either a positive integer (int) or a positive floating point (float).

**Expression E** is an integer, a floating point, an identifier, or an infix arithmetic expression with operators "+" and “\*” only. These two operators are left associative (e.g., 1 + 2 + 3 is equivalent to (1 + 2) + 3). “\*"

has higher precedence than “+”.

**Id = E** assigns the value of an expression E to the variable Id. **printvar Id** outputs the value of Id.

If there is any **syntax error**, you are expected to interpret the program until the statement where you find the error. Also, **your error message must contain the line number where the error was found.**

Tokens may be separated by **any number of** white spaces, tabs, or new lines.

**Type checking rules are given below:**

Stmt -> int Id | {Id.type = 0}

float Id | {Id.type = 1}

Id = E {if (Id.type \= E.type) then type error}

E 🡪 Integer | {E.type = 0}

Float | {E.type = 1}

Id | {E.type = Id.type}

E1 + E2| {if (E1.type==E2.type) then E.type = E1.type; else type error}

E1 \* E2 | {if (E1.type==E2.type) then E.type = E1.type; else type error}

If one of the above rules is violated, your program should terminate and print “<line number>: type error”. In addition, if a variable is used but is not declared, then your program should print “<line number>: <variable name> is used but is not declared”.

***Compile your program:***

flex –l calc.l

bison -dv calc.y

gcc -o calc calc.tab.c lex.yy.c –lfl

***Execution (example):***.

**./calc < input**

Where **input** is the name of the input file

---------------------------------------------------------------------------------------------

***Example Programs (Note that the test cases used in grading may be different from the examples)***

---------------------------------------------------------------------------------------------

**Program 1:**

main() {int x; x = 3;}

**Output:**

---------------------------------------------------------------------------------------------

**Program 2:**

main() {int x; x = 3; printvar x;}

**Output:**

3

---------------------------------------------------------------------------------------------

**Program 3:**

main() {int x; x = 3; x = x + 4; printvar x;}

**Output:**

7

---------------------------------------------------------------------------------------------

**Program 4:**

main() {x=3;}

**Output:**

Line 1: x is used but is not declared

---------------------------------------------------------------------------------------------

**Program 5:**

main() {int x; x = 1-2; }

**Output:**

Parsing error: line 1

---------------------------------------------------------------------------------------------

**Program 6:**

main() {

int 1x;

}

**Output:**

Parsing error: line 2

---------------------------------------------------------------------------------------------

**Program 7:**

main() {

int x;

x = 3;

printvar x;

}

**Output:**

3

---------------------------------------------------------------------------------------------

**Program 8:**

main() {

float x; x = 1.2; printvar x;

}

**Output:**

**1.2**

---------------------------------------------------------------------------------------------

**Program 9:**

int x;

**Output:**

Parsing error: line 1

---------------------------------------------------------------------------------------------

**Program 10:**

main() {int x; int y; x = 3; y = x; printvar y;};}

**Output:**

3

---------------------------------------------------------------------------------------------

**Program 11:**

main() {int x; float y; x = 1; y = x;}

**Output:** Line 1: type error

---------------------------------------------------------------------------------------------

**Program 12:**

main() {int x; x = 3 + 1.5; printvar x;}

**Output:** Line 1: type error

---------------------------------------------------------------------------------------------

**Program 13:**

main() {printvar 1 + 2;}

Output: Parsing error: Line 1

---------------------------------------------------------------------------------------------