

# Assignment 1

**Due: 11:59pm September 23 (Sunday)**

**This assignment is done individually or by a group of 2 students**

- Each group should submit only ONE copy of the assignment
- Please include the name, the section #, and the email of all group members in the readme file.

## 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 programs whose syntax is defined below.**

Prog  $\rightarrow$  main() {Vardefs; Stmts}  
Vardefs  $\rightarrow$   $\epsilon$  | Vardef; Vardefs  
Vardef  $\rightarrow$  int Id | float Id  
Stmts  $\rightarrow$   $\epsilon$  | Stmt; Stmts  
Stmt  $\rightarrow$  Id = E | **printID** Id | **printExp** E  
E  $\rightarrow$  Integer | Float | Id | E - E | E \* E  
Integer  $\rightarrow$  digit+  
Float  $\rightarrow$  Integer . Integer | Integer "E" Integer | Float "E" Integer

**Prog** defines a program that contains only one function main().

**Vardefs** is a sequence of variable declarations. A program may or may not have variable declarations.  $\epsilon$  specifies empty variable declarations. Each variable Id is either a positive integer (int Id) or a positive floating point (float Id).

- A positive integer is a sequence of digits from 0 to 9, e.g. 2, 96.
- A positive floating point number is a decimal point (e.g. 2.16), or an integer/decimal point followed by an optional integer exponent part (e.g., 1.5E2). The character 'E' separates the mantissa and exponent parts. E.g. 1.5E2 ( $1.5 \cdot 10^2$ ) and 2E5 ( $2 \cdot 10^5$ ) are valid floating points.

**Stmts** is a sequence of statements. A program may or may not have statements.  $\epsilon$  specifies empty statement.

**Id** is an identifier, which starts with a lower-case letter and followed by 0 or more lower-case letters, capital letters, or digits. For example, x, x1, xy, xY, x12Z are identifiers, but 1x and A1 are not. **int Id** defines an integer variable. **A new integer variable gets 0 as its initial value.** **float Id** defines a floating point variable. **A new floating point variable gets 0.0 as its initial value.**

**Expression E** is a floating point, an identifier, or an infix arithmetic expression with operators "-" (subtraction) and "\*" (multiplication) 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. **printID Id** prints the value of Id. **printExp E** prints the value of an expression E.

If an input does not match any token, output "lexical error: <input>", where <input> is the input.

If there is a **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:**

Vardef $\rightarrow$ int Id	{Id.type = INT}
float Id	{Id.type = FLOAT}
Stmnt $\rightarrow$ Id = E	{if (Id.type $\neq$ E.type) then type error}
E $\rightarrow$ Integer	{E.type = INT}
Float	{E.type = FLOAT}
Id	{E.type = Id.type}
E1 - E2	{if (E1.type $\neq$ E2.type) then E.type = E1.type; else type error}
E1 * E2	{if (E1.type $\neq$ E2.type) then E.type = E1.type; else type error}

If one of the 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:*

- (1) the test cases used in grading may be different from the examples given below*
- (2) you can assume that different variables have different names in the programs*
- (3) You do not need to consider nested blocks – there is only one block (the main function)*

---

**Program 1:**

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

**Output:**

---

**Program 2:**

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

**Output:**

3

---

**Program 3:**

```
main() {int x; x = 20; x = x - 2 * 5; printfID x;}
```

**Output:**

10

---

**Program 4:**

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

**Output:**

Line 1: x is used but is not declared

---

**Program 5:**

```
main() {int x; x = 1+2; }
```

**Output:**

Lexical error: +  
Parsing error: line 1

---

**Program 6:**

```
main() {int 1x;}
```

**Output:**

Lexical error: 1x  
Parsing error: line 1

---

**Program 7 (new line, tabs, spaces):**

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

**Output:**

3

---

**Program 8:**

```
main() {float x; x = 1.2; printfID x;}
```

**Output:**

1.2

---

**Program 9:**

```
main() {float x; x = 1E2; printfID x;}
```

**Output:**

100.00

---

**Program 10 (a floating point gets initial value 0.0):**

```
main() {float x; printID x;}
```

**Output:**

0.0

---

**Program 11 (no main function):**

```
int x;
```

**Output:**

Parsing error: line 1

---

**Program 12:**

```
main() {int x; int y; x = 3; y = x; printID y;}
```

**Output:**

3

---

**Program 13 (assigning an integer to a float) :**

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

**Output:** Line 1: type error

---

**Program 14 (an integer subtract a floating point):**

```
main() {int x; x = 3 - 1.5; printID x;}
```

**Output:** Line 1: type error

---

**Program 15:**

```
main() {printExp 5-2;}
```

Output: 3

---

**Program 16 (no variable declarations, no statement):**




```
main() { }
```

**Output:**

### Submission guideline

- Please hand in your **source code** and a **Makefile** electronically (**please do not submit .o or executable code**). You must make sure that your code compiles and runs correctly on bingsuns.binghamton.edu. The Makefile **must** give the executable code the name **calc**
- Write a **README** file (**text file, please do not submit a .doc file**) which contains
  - The name, the section, and the email address of group members
  - Whether your code was tested on bingsuns or remote.cs. If your program does not work on bingsuns and remote.cs, but work on your local machine, please show demo to TA.
  - How to execute your program.
  - (optional) Briefly describe your algorithm or anything special about your submission that the TA should take note of.
- Place all your files under one directory with a unique name (such as p1-[userid] for assignment 1, e.g. p1-pyang).
- Tar the contents of this directory using the following command.  
**tar -cvf [directory\_name].tar [directory\_name]**  
E.g. tar -cvf p1-pyang.tar p1-pyang/
- Use mycourses to upload the tared file you created above.
- 

### Grading guideline

-  Readme (must be a text file), correct executable names: 4'
-  Correct makefile (all files are compiled when typing **make**): 8'
-  Correctness of the program: 88'

### Academic Honesty:

All students should follow **Student Academic Honesty Code** (<http://watson.binghamton.edu/acadhonor-code.html>). All forms of cheating will be treated with utmost seriousness. You may discuss the problems with other students, however, you must write your OWN codes and solutions. Discussing solutions to the problem is NOT acceptable. Copying an assignment from another student or allowing another student to copy your work may lead to an automatic **F** for this course. If you borrow small parts of code/text from Internet, you must acknowledge this in your submission. Also, you must clearly understand and be able to explain the material. Copying entire material or large parts of such material from the Internet will be considered academic dishonesty. **Moss will be used to detect plagiarism in programming assignments.** You need ensure that your code and documentation are protected and not accessible to other students. Use **chmod 700** command to change the permissions of your working directories before you start working on the assignments. If you have any questions about whether an act of collaboration may be treated as academic dishonesty, please consult the instructor before you collaborate.