Assignment #3

1. Make an interpreter for a calculator

간단한 사칙 연산을 할 수 있는 계산기를 lex / yacc 를 사용하여 작성하라.

- 작성한 계산기 프로그램은 여러 개의 expression을 받아 들인다.
  - 각각의 expression은 <return key>로 분리된다.
  - 프로그램은 expression을 하나 받아 들이면, 바로 그 결과 값을 print 한다.
  - 프로그램은 사용자가 <CTRL-D> (또는 EOF) 를 입력하면 종료한다.
- 입력하는 숫자는 정수(integer)와 실수(double precision floating point number)의 2가지 타입이 있다.
  - 정수끼리의 연산은 항상 정수 값을 돌려 주어야 한다.
  - 실수끼리 연산하거나, 정수와 실수를 연산하면, 실수 값을 돌려 주어야 한다.
  - 실제 계산은 우선 정수 값을 실수로 conversion 한 후, 실수끼리 연산하도록 한다.
  - 실수끼리의 modulo 연산은 fmod( ) 함수를 참고하라.
- 단, 실수끼리 연산이거나, 정수와 실수 연산이더라도, 비교 연산자(<, >, <=, >=, ==, !=), 논리 연산자(&&, ||, ! 등)인 경우는 정수 값 0 또는 1을 돌려 주어야 한다.
  - 예를 들어, 34.3 > 10.1 의 연산 결과는 정수 1 이다.
- 비교 연산자, 논리 연산자에서는 0 을 false 로, 1 을 true 로 사용한다.
- 지원해야 되는 연산자는 +, -의 unary operator, +, -, *, /, % 등의 산술 연산자, <, >, <=, >=, ==, != 등의 비교 연산자, &&, ||, ! 등의 논리 연산자, = 의 대입 연산자 등이다. 이 연산자 부분의 syntax는 tiny language의 expression 부분과 거의 일치한다. 단, tiny language와는 달리, 배열과 함수는 지원하지 않는다.
- 변수도 지원하여야 한다. 변수에 대입하면, 대입한 값이 저장되어야 하고, 변수 명을 쓰면, 미리 저장되었던 값이 나와야 한다. 또, 변수의 타입도 저장된 타입이 그대로 나와야 한다. 단, 저장되지 않은 변수 명을 쓰면, integer 0을 돌려준다. 이를 위해서는 간단한 symbol table 관리 routine이 필요할 것이다.
- 전체적으로, assignment #2에서 작성한 lex, yacc 프로그램에서 불필요한 부분을 제거하면, starting point가 될 것이다.

완성된 yacc/bison 프로그램과 lex/flex 프로그램은 각각 c03B_<자기학번>.y 와 c03C_<자기학번>.l 로 저장하라.

- 전체 프로그램에서 공통으로 사용하는 type이나, function 등은 별도의 header file c03A_<자기학번>.h 에 정의할 수 있다. (불필요하면, 정의하지 않아도 된다.)
- c03A_<자기학번>.h 에서 정의한 function들은 별도의 C 파일을 만들지 말고, c03B_<자기학번>.y 의 뒷부분에서 구현하도록 한다.
- c03B_<자기학번>.y 에서 만들어내는 y.tab.h 는 이름을 c03B_<자기학번>.h 저장하고, c03C_<자

기학번>.l 에서는 #include 시에 #include "c03B_<자기학번>.h" 형태를 쓰도록 한다. (채점의 편의를 위해, 꼭 이렇게 해 주기 바람)

- main( ) function은 반드시 c03B_<자기학번>.y 에 정의하도록 한다.
- 최종적으로, 총 4개의 파일, c03A_<자기학번>.h, c03B_<자기학번>.y, c03B_<자기학번>.h, c03C_<자기학번>.l 을 제출하여야 한다.

작성한 프로그램은 전체적으로 다음과 같이 작동하여야 한다. (굵은 글씨는 사용자 입력, 분홍 글씨는 참고용 설명, '$' 는 prompt, 아래 예제는 Unix/Linux host에서의 실행 예제이다.)

```
$ ls -l c03*.*
-rwxr-xr-x      1 Administ 없음             671 Nov   9 22:11 c03A.h*
-rwxr-xr-x      1 Administ 없음             253 Nov   9 22:09 c03Atest.txt*
-rw-r--r--      1 Administ 없음            2004 Nov   9 21:17 c03B.h
-rwxr-xr-x      1 Administ 없음            5043 Nov   9 21:17 c03B.y*
-rwxr-xr-x      1 Administ 없음            1026 Nov   9 21:17 c03C.l*
$ yacc -d -o c03B.c c03B.y
$ flex -oc03C.c c03C.l
$ ls -l c03*.*
-rwxr-xr-x      1 Administ 없음             671 Nov   9 22:11 c03A.h*
-rwxr-xr-x      1 Administ 없음             253 Nov   9 22:09 c03Atest.txt*
-rw-r--r--      1 Administ 없음           39494 Nov   9 22:58 c03B.c
-rw-r--r--      1 Administ 없음            2004 Nov   9 22:58 c03B.h
-rwxr-xr-x      1 Administ 없음            5043 Nov   9 21:17 c03B.y*
-rw-r--r--      1 Administ 없음           39983 Nov   9 22:58 c03C.c
-rwxr-xr-x      1 Administ 없음            1026 Nov   9 21:17 c03C.l*
$ gcc -o c03A.exe c03B.c c03C.c -lfl
$ c03A.exe
12 + 24
36
12.1 + 24.3
36.400000
12.1 * 2
24.200000
<CTRL-D>
$ cat c03Atest.txt
103 + 20
103 + 20.45
103.45 + 20
103.40 + 20.05
103 - 20
103.40 - 20.05
20 * 4.5 / 2
20 % 3
20.5 % 3
10 < 20
10.5 > 20
0 == 0.0
1 != 0
false = 0
true = 1
true && false
true || false
! true
((a = 1 + 2 * 3 / 4.1 - -2) < 10) == true
a
```

```
$ c03A.exe < c03Atest.txt
123                     <-- 103 + 20
123.450000              <-- 103 + 20.45
123.450000              <-- 103.45 + 20
123.450000              <-- 103.40 + 20.05
83                      <-- 103 - 20
83.350000                       <-- 103.40 - 20.05
45.000000                       <-- 20 * 4.5 / 2
2                       <-- 20 % 3
2.500000                <-- 20.5 % 3
1                       <-- 10 < 20
0                       <-- 10.5 > 20
1                       <-- 0 == 0.0
1                       <-- 1 != 0
0                       <-- false = 0
1                       <-- true = 1
0                       <-- true && false
1                       <-- true || false
0                       <-- ! true
1                       <-- ((a = 1 + 2 * 3 / 4.1 - -2) < 10) == true
4.463415                <-- a
$
```

_____ 끝 _____