Assignment #2

1. Make a scanner using lex/flex

컴파일러 과목의 실험용으로, tiny language를 정의한다. tiny language의 token들에 대한 정의는 별도 문서에 제시되어 있다. tiny language의 token 정의를 참고하여, 각 token을 인식하고, 각 token의 type과 value를 출력하는 lex (또는 flex) 프로그램을 작성하라.

각 type 별 처리는 다음과 같다.

- delimiter : 일체의 출력을 하지 않는다.
- decimal constant, octal constant, hexadecimal constant : type은 INTVAL, value는 대응되는 10진수 값
- floating point constant : type은 FLOATVAL, value는 대응되는 C/C++의 double 값
- string constant : type은 STRVAL, value는 앞뒤의 " (double quote)를 제외한, 그 사이의 문자열
- identifier : type은 IDEN, value는 대응되는 문자열
- keyword : type은 KEYWORD, value는 대응되는 문자열 (예: if --> (KEYWORD, if))
- comment : 일체의 출력을 하지 않는다.
- operator : type은 OPER, value는 대응되는 문자열

작성한 lex / flex 프로그램은 별도로 지시하는 방법에 따라 제출하라.

저장한 프로그램은 lex 로 compile 한 후, output 된 C program을 compile 하면, 다음과 같이 작동하여야 한다. (굵은 글씨는 사용자 입력, '$' 는 prompt, 아래 예제는 Unix/Linux host에서의 실행 예제이다.)

```
$ cat __example__.cpp
/* 아래는 C++ program이지만, 최종적으로 tiny compiler가 같은 결과를 내게 됨. */
#include <iostream>
#include <string>
#define print(x)        std::cout << x
#define println(x)      std::cout << x << std::endl

/* this is a C-style comment */
int main(void) {
        int i;
        float x;
        println("hello world");
        i = 0;
        i = 1234;
        i = 01234;
        i = 0x1234;
        print("i = "); println(i);
        x = 12.;
        x = 12.34;
```

```
        x = .34;
        x = 12.34E+2;
        print("x = "); println(x);
        return 0;
}
$ flex -omain.c main.l
$ gcc -o main.exe main.c -lfl
$ main.exe < __example__.cpp
KEYWORD: int
IDEN: main
OPER: (
KEYWORD: void
OPER: )
OPER: {
KEYWORD: int
IDEN: i
OPER: ;
KEYWORD: float
IDEN: x
OPER: ;
IDEN: println
OPER: (
STRVAL: hello world
OPER: )
OPER: ;
... (상당한 출력 생략) ...
KEYWORD: return
INTVAL: 0
OPER: ;
OPER: }
$
```

끝