```
/* recognize tokens for the calculator and print them out */
/*
Flex规则文件的可选部分，用于包含C代码段。在这个部分中，定义了一个枚举类型 enum yytokentype，用于表示不同标记的类型。例如，NUMBER 表示数字，ADD 表示加号，SUB 表示减号，以此类推。还定义了一个整数 yylval 用于存储标记的附加信息（例如数字的值）。
*/
%{
enum yytokentype {
    NUMBER = 258,
    ADD = 259,
    SUB = 260,
    MUL = 261,
    DIV = 262,
    ABS = 263,
    EOL = 264
};
int yylval;
%}


/*
用于分隔规则定义和处理规则定义

这个部分中，使用正则表达式规则来定义如何识别不同的标记。每个规则由正则表达式和与之相关联的操作组成。
"+" { return ADD; } 规则表示当输入中出现一个加号时，返回标记 ADD。
[0-9]+ { yylval = atoi(yytext); return NUMBER; } 规则表示当输入中出现一个或多个数字时，将数字的值存储在 yylval 中并返回标记 NUMBER。
"\n" { return EOL; } 规则用于识别换行符。
[ \t] { ... } 规则用于忽略空格和制表符。
. 规则用于捕获任何未匹配到其他规则的字符，并输出一个包含这些字符的 "Mystery character" 消息。
*/
%%
"+" { return ADD; }
"-" { return SUB; }
"*" { return MUL; }
"/" { return DIV; }
"|" { return ABS; }
[0-9]+ { yylval = atoi(yytext); return NUMBER; }
\n  { return EOL; }
[ \t] { /* ignore whitespace */ }
.   { printf("Mystery character %c\n", *yytext); }
%%


/*
main 函数：

这是C代码部分，主要用于测试和运行词法分析器。它包含了一个无限循环，不断调用 yylex() 来读取标记。
每次调用 yylex() 时，它会返回一个标记的类型，该类型会被打印出来。
```

```
如果标记是 NUMBER，它还会打印附加信息（即 yylval 中存储的数字的值）。
当标记是换行符时，它会打印一个新行，表示结束。
*/
main(int argc, char **argv)
{
int tok;

    while(tok = yylex()) /*yylex()内置的读取输入函数*/
    {
        printf("%d", tok);
        if(tok == NUMBER)
            printf(" = %d\n", yylval);
        else
            printf("\n");
    }
}
```