一，已实现功能：

全部必做及全部选做

即识别词法错误、语法错误及识别八进制和十六进制数，识别指数形式的浮点数，识别“//”和“/*...*/”形式的注释。

整体上，采用多叉树的形式建立语法书，每个语法单元及词法单元都是语法树的一个节点。在`lexical.l`中匹配所有词法单元，建立相应节点并返回`token`，在`syntax.y`中利用产生式匹配并建立语法树。

定义了结构体Node作为树的节点：

```
struct Node{
    int lineno, childNum, type;//行号、子节点数目、节点的类型
    char* token, *val;//词法单元/语法单元名、单元的值（没有则为NULL）
    struct Node** child;//指向子节点
};
```

函数`void printError(int lineno, int type, int error, char* msg);`用于输出错误信息，其中`type`用于区分A类型还是B类型的错误，error则表明是哪种具体的错误，如八进制识别错误、缺少字符、出现未定义符号等等（不过到最后因为麻烦把好多错误都归为了语法错误`syntax error`）。为了满足每行只输出一个错误，定义变量`error_line`指向目前出错的行号，如果该行已有错误输出，则余下的错误都不会输出

函数`pNode createNode(int lineno, int type, char* token, char* val, int childNum, ...)`用于创建树的节点

由于我们不知道某个节点会有多少个子节点，因此用到了变长参数，采用`va_list`来获取一系列参数。

函数`void printTree(pNode root, int level);`用于打印语法树，level为节点的层数，用于标记该节点需要缩进的空格数

词法错误：

在所有匹配的末尾加上通配符“.”的匹配，并输出词法错误匹配失败的提示信息

最初写词法错误时发现，如果出错的地方没有返回而直接终止，会导致余下的词法错误无法被检测到，如下。

```
int main()
{
     int i = 09;
     int j = 0x3G;
}
```

因此在检测到错误的数字时，返回一个数值为0的`INT`型数字，以便程序可以继续分析

语法错误：

```
Stmt → error SEMI
CompSt → error RC
Exp → error RP
```

利用如图的产生式进行匹配及错误恢复，以便能继续匹配余下的文本内容

八进制识别：

识别出数字首位的0后，采用循环的方式依次给每位的数字乘8。同时，支持形如支持00011的输入

十六进制识别：

识别出数字起始的0x后，采用循环的方式依次给每位的数字乘16。同时，支持形如支持0x0011的输入

识别指数形式浮点数：

即匹配浮点数+e+正负号+无符号整数

识别“//”和“/.../”形式的注释：

直接用“//”与“/*”作为正则表达式进行文本匹配

二，编译方法

在`Code`文件夹下写有`Makefile`文件，通过执行`make`生成parser文件，执行`make test`使用`parser`对`test.cmm`文件进行词法和语法分析。也可直接通过`bash run.sh`命令，自动顺序执行`make` `make test` `make clean`三个命令。

三，实验感想

最困惑的是有时候我明明通过`%nonassoc`规定了优先级，但是依然会有规约/规约或者/移入/规约的问题，不知如何解决。这个错误恢复是真的烦，动不动就会写出来一堆二义性冲突，令人伤心。而且有时候前面的错误处理不好会导致后面的错误恢复全部出错如：

```
compiler@DESKTOP-3CKCV4M:~/lab1/Code$ make test
./parser test.cmm
int a[100]Error type B at line 1: ]
;
```

```
VarDec :ID %prec LOWER_THAN_LP{$$ = createNode(@$.first_line,grammer_,"VarDec",NU
    | VarDec  LB  INT  RB {$$ = createNode(@$.first_line,grammer_,"VarDec",NULL,4
    | VarDec  LB  error RB {findError=1; printError(yylineno,B_,func_,yytext);}
    | VarDec  error  INT  RB {findError=1; printError(yylineno,B_,char_,"[");}
    ;
```

终于写完了，呼~