心得体会

一. 文法的预测分析表：

	a	+	*	(	)	#
E	->TE`			->TE`		
E`		->+TE`			->ε	->ε
T	->FT`			->FT`		
T`		->ε	->*FT`		->ε	->ε
F	->a			->(E)		
#						Acc

二. 心得体会

1. 何元梅：

这次实验还算比较简单，一开始看到这个题目，我首先的思路是先申请两个栈，一个存放输入串，一个存放非终结符，也就是分析栈，值得注意的是两个栈在入栈的时候都是逆序进栈的，然后根据所给的文法产生式采用递归推下去。理清思路后，写出 main 函数，给出大概的程序构架，然后分派给组员相应的任务，每个人负责相应的板块，最后衔接在一块。在实现的过程中，有组员想到可以不用产生式，这样每个函数的所要执行的判断条件遗漏，不容易考虑周全，可以直接用预测分析表可以一眼看出每个子程序所要执行的步骤机器控制条件，于是我们先在草稿本上求出所给文法的 FIRST 集，FOLLOW 集和 SELECT 集,然后列出了所给文法的预测分析表，直接根据表格的内容完成了所有子程序，速度很快，接下来我们就开始了综合调 bug 的时间了，首次上传上去只有两个测试样例通过测试，于是我们就边根据文法推出不同情况的测试样例以便测试我们是哪个模块没有考虑周全或者程序的健壮性存在哪些问题，后来我们发现我们不同函数之间衔接存在一些问题，存放同一个字符的时候前后的两个函数都对它进行了操作，于是就输出了错误信息，我们发现之后就统一在一个函数里处理，调试的过程还是比较很开心的，由于后来我们对修改的方法不统一，每个人都自己的想法，于是就每个人独立去修改，最后选择一个我们综合起来最优的版本。有一种对自己的作品检验的感觉，有点成就感，最后我们对该版本测试了尽可能多的不同情境下的测试样以确定它的健壮性。

通过这次实验，使我们更加熟悉 LL(1)文法，加深了对 LL(1)文法的认识，增强了团队合作意识和合作能力，合作起来很愉快。

2. 孙程程：

这次上机实验不是特别难，开始根据老师发的上课 ppt 里提供的思路，对每句文法分析利用函数循环递归嵌套分析处理每个非终结符，发现对其中一些情况考虑不周到，后来我们采取根据文法先分别求 first 集和 follow 集，然后求得 select 集，最后得到分析表，建立分析栈存储非终结符，再建立一个栈存储输入串，根据分析

表中每个非终结符和终结符对应关系处理两个栈里栈顶元素入栈出栈操作。整个思路理清楚后，函数很快就写好了，然后我们就遇到了第一个难题，对于每个输入串的处理结束如何跳出所有循环，直接在函数输出错误或者正确的时候使用 return 返回，就遇到了一个函数里连续引用了两个函数，第一个函数结束以后还会去执行下一个函数，就会出现多行输出，最后我们选择设置全局变量 flag 来判断需不需要走下一个函数，如果第一个函数返回错误，那么 flag 等于 0，再进入下一个函数的时候先判断 flag，由于 flag=0，就直接不进入下一个函数，返回主函数，输出信息，正确情况也是如此。设置完返回值问题后，整个代码已经可以正常运行，我们就开始了调 bug 的过程，首先是对于括号的处理，由于我们每个人负责的函数的处理不一样，而且我开始对于分析表的理解不到位，在 T`遇到右括号时我直接把右括号处理了，而在另一团队成员 F 函数里也对左右括号进行了处理，导致出现错误，由于 T`可推到空，所以对于遇到右括号的情况，直接在分析栈里弹出 T'即可，修改以后可以完成操作。我们还遇到了+号不能处理等其他情况，我们引用'了输出函数挨个输出两个栈的取出的栈顶元素和栈里剩余元素，挨个分析整个分析过程，修改函数最终得到了完整代码。

这次实验最终能够成功，主要是思路比较清晰，代码写起来就很流畅，这说明以后更需要好好听课，课上捋顺思路，课下才能更好完成作业。

3. 陈智瑞：

本次实验的目的是验证平台输入的测试样例是否是 LL1 文法，我们要做的就是根据题目给出的语法设计出每种情况下要下一步到哪里处理，尽最大可能的想出所有情况和每个分函数的具体流程。

值得庆幸的是，老师考虑到我们现阶段的实际水平，给出的相关语法与课本上的例题一致，因此我们整个编码的流程就是根据课本 94 页上的表格编写的。主要的设计原理就是申请两个栈，分别是分析栈和写入栈，开始时将开始符#和起始 E 压入分析栈，同时将#压入写入栈。考虑到栈先进后出的特点，为了方便处理，将输入的测试样例倒序入栈。待写入栈输入完毕后，开始进行分析，首先进入 Produce_E 函数，根据写入栈顶元素的不同再传送到不同文法对应的函数中，多函数递归嵌套使用，最后得出结果。

我们后期主要纠结的问题是如果已经判断出来测试样例不属于 LL1 文法，得出其错误的结果后怎样让函数停止递归，直接跳出，得到正确的输出结果。我决定设置一个全局变量来解决此问题，其初始值为 1，如果在任何一个分函数里判断出测试样例不是 LL1，就将全局变量的值改为 0，并利用 return 跳出函数。我选择在 main 函数中统一输出，根据全局变量的值来输出相应的语句，进而避免了函数嵌套太多导致多行输出的后果。

这个想法的确不错，我也得到了少数测试样例的正确输出，但是在 CG 平台上提交后不出意外的不是全对，这就说明代码中的某个函数存在漏洞。经过我绞尽脑汁的测试后，终于试出来了一个不符合要求的输出结果。经过多次的逐步调试后以及对比后，最终找到了错误的原因：在处理 F 式的函数 Produce_F 中，忘了最起始的分析栈弹出栈顶操作，导致出错，经过修改后，终于得出了正确结果。

4. 毛萍兰：

本次 LL1 文法编程作业整体还是比较简单的，把每一个非终结符对应的表达式作为一个函数写出来，然后相互调用就可以完成了。其中遇到的第一个问题在于，假如识别结果已经发生错误，如何让对字符串的扫描停止并跳出所有函数。一开始

的方法是，在所有该报错的地方 return，并且输出报错信息，但这样造成会使得上一层嵌套无法跳出，继续报错很多行。所以最后采用了全局变量 flag 来判断是否出错，进入函数前先判断当前的 flag 是否为 1，即目前没有发生错误，如果为 1 则进入函数否则不进入函数并且结束。最后在主函数判断，如果识别完毕后 flag=1 则正确，反之错误，输出相应的信息。修改完这个部分的问题，当测试 a)这个字符串时，把结果应该是判断为错，但是却判断为正，于是编辑了一个函数输出目前分析栈和符号栈的内容，放在每个函数的第一行，结果发现在分析栈为＃时就对符号串的分析就结束了，而此时的符号串还是)＃，由于不进入任何函数，所以也无法报错，为了解决这个问题，在主函数部分加了一个条件判断，即函数嵌套结束的时候，如果分析串栈顶元素为＃号而字符串栈顶元素不为＃，则说明字符串不是该 LL1 文法的字符串，flag=0，成功的解决了这个问题。随后又测试了一系列有+的字符串，有*号的字符串，有括弧的字符串。发现对非终结符取空的这个条件把握的是很准确，原函数里为肥+和*就直接报错了，其中包括了空，但是实际上，当我们分析字符串的时候，目前分析串的栈顶元素取了空值则直接出栈并跳到下一个分析栈顶与字符串栈顶的比对，所以，非终结符能取空的两个表达式里，若没遇到+和*应该是不报错的，没有错误的。修正完毕以后，所有测试过的关于+ * 和括弧的表达式终于得到了正确的判断结果。

本次编程过程还是挺有意思的，虽然四人组队，但每个人写一两个函数，拿到最初那一版整合的代码后，面对上述问题的解决方法和思路又完全不同，所以四个人每人按照自己的思路对整个函数进行修改，结果是大家互不相同，但最后四个人的版本的函数终于都能在 cg 平台上获得测试样例都正确的版本。最终我们选择了一个运行时间最短的版本作为本此群作业上交的最终版本。