（若发现问题，请及时告知）

A1 给定文法 $G[S]$：

$S \to AB$
$A \to aA$
$A \to \varepsilon$
$B \to bB$
$B \to \varepsilon$

针对文法 $G[S]$，下表给出各产生式右部文法符号串(*rhs(r)*)的 *First* 集合，各产生式左部非终结符(*lhs(r)*)的 *Follow* 集合，以及各产生式的预测集合 *PS*。试填充其中空白表项的内容(1)-(6)，验证该文法是否 LL(1)文法（说明原因），并补全基于该文法构造的递归下降分析程序(7)-(10),其中 getToken()为取下一单词的过程，变量 lookahead 存放当前单词。

G 中的规则 r	*First* (*rhs*(*r*))	*Follow* (*lhs*(*r*))	*PS* (*r*)
$S \to AB$	（1）	#	（2）
$A \to aA$	*a*	（3）	*a*
$A \to \varepsilon$	ε	此处不填	（4）
$B \to bB$	（5）	#	（6）
$B \to \varepsilon$	ε	此处不填	#

```
void ParseS( )                    // 主函数
{
    ParseA( );
    ParseB( );
}
void ParseA( )
{
    switch (lookahead)            // lookahead 为下一个输入符号
    {
        case ′ a′ :
              （7）;
            ParseA();
            break;
        case （8）:
            break;
        default:
            printf("syntax error \n")
            exit(0);
```

```
    }
    return A_num;
}
void ParseB( )
{
    switch (lookahead)  {
         case  （9）  :
             MatchToken( 'b' );
               （10）  ;
             break;
         case ' #' :
             break;
         default:
             printf("syntax error \n");
             exit(0);
    }
}
void  Match_Token(int expected)
{
    if (lookahead != expected)
    {
             printf("syntax error \n")
             exit(0);
    }
    else
         lookahead = getToken();
}
```

A2 补全下列文法（与上一题相同）的分析表，根据分析表验证该文法是否 LL(1)文法（说明原因），并补全输入符号串 *aabb* 的表驱动 LL(1)分析过程（步骤 3、6、9）。

$S \to AB$
$A \to aA$
$A \to \varepsilon$
$B \to bB$
$B \to \varepsilon$

	a	*b*	#
S	$S \to AB$	（1）	$S \to AB$
A	（2）	（3）	$A \to \varepsilon$
B	*此处不填*	（4）	$B \to \varepsilon$

步骤	*下推栈*	余留符号串	*下一步动作*
1	*#S*	*aabb#*	*应用产生式* $S \to AB$
2	*#BA*	*aabb#*	*应用产生式* $A \to aA$
3			
4	*#BA*	*abb#*	*应用产生式* $A \to aA$
5	*#BAa*	*abb#*	*匹配栈顶和当前输入符号*
6			
7	*#B*	*bb#*	*应用产生式* $B \to bB$
8	*#Bb*	*bb#*	*匹配栈顶和当前输入符号*
9			
10	*#Bb*	*b#*	*匹配栈顶和当前输入符号*
11	*#B*	*#*	*应用产生式* $B \to \varepsilon$
12	*#*	*#*	*返回，分析成功*

A3 给定命题表达式文法 $G[S]$：

$$S \to P$$
$$P \to \wedge P\ P \mid \vee P\ P \mid \neg P \mid id$$

其中，$\wedge$、$\vee$、$\neg$ 分别代表命题逻辑与、或、非等运算符单词，*id* 代表标识符单词。

容易得出: $G[S]$ 是 $LL(1)$ 文法。基于$G[S]$的预测分析表和一个分析栈，课程中介绍了一种表驱动的 $LL(1)$ 分析过程。假设有输入符号串：$\vee \vee a \wedge b\ c \vee \neg a \wedge c\ b\ \#$。试问，在分析过程中，分析栈中最多会出现几个$S$？几个$P$？若因误操作使输入串多了一个符号，变为 $\vee \vee a \wedge b\ c\ c \vee \neg a \wedge c\ b$，当分析过程中发生错误时，关于报错信息，你认为最不可能的选择是（4选1）：（1）缺运算数；（2）多运算数；（3）缺运算符；（4）多运算符。如果想要从该出错位置恢复分析，可以进行什么操作？

以下是 Lecture03 文档中的题目

A4 设有文法 $G[S]$：

$$S \rightarrow aSb \mid aab$$

若针对该文法设计一个自顶向下预测分析过程，则需要向前察看多少个输入符号？

A5 试验证下列文法 G [S] 是 LL(1) 文法:

$$S \rightarrow P \mid \varepsilon$$

$$P \rightarrow (P)P \mid a$$

其中 （，），以及 a 为终结符

A6 给出如下文法的预测分析表，并根据分析表指出相应文法是否 LL(1)文法。

$$S \rightarrow aSa \mid bSb \mid \varepsilon$$

A7 按照本讲介绍的消除一般左递归算法消除下面文法 $G[S]$中的左递归（要求依非终结符的排序 S、Q、P 执行该算法）：

$$S \rightarrow PQ \mid a$$
$$P \rightarrow QS \mid b$$
$$Q \rightarrow SP \mid c$$