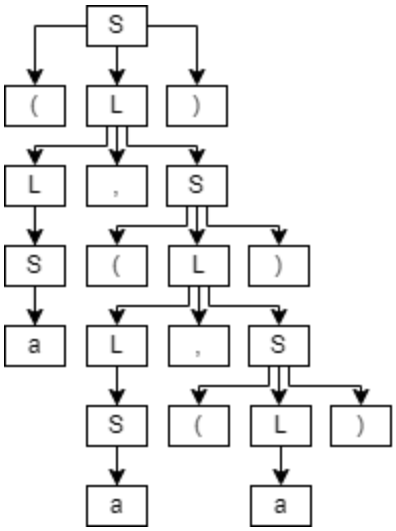
Homework 3

（1）针对习题3.1文法，给出(a,(a,(a)))的最左推导、最右推导（并用下划线标出右句型句柄）和分析树。

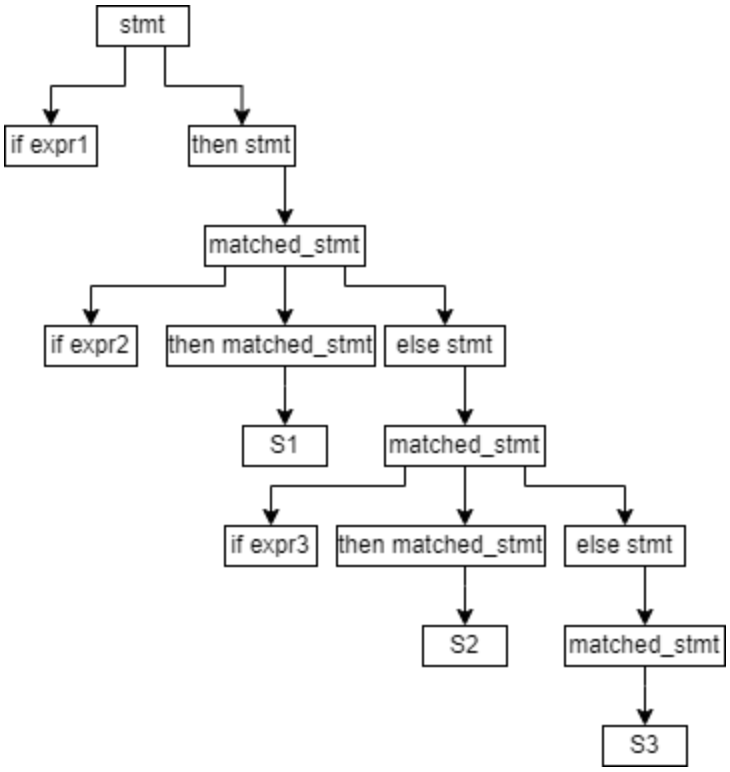
最左推导：$S \Rightarrow \underline{(L)} \Rightarrow (\underline{L,S}) \Rightarrow (\underline{S},S) \Rightarrow (\underline{a},S) \Rightarrow (a,\underline{(L)}) \Rightarrow (a,(\underline{L,S})) \Rightarrow (a,(\underline{S},S)) \Rightarrow (a,(\underline{a},S)) \Rightarrow (a,(a,\underline{(L)})) \Rightarrow (a,(a,(\underline{S}))) \Rightarrow (a,(a,(\underline{a})))$

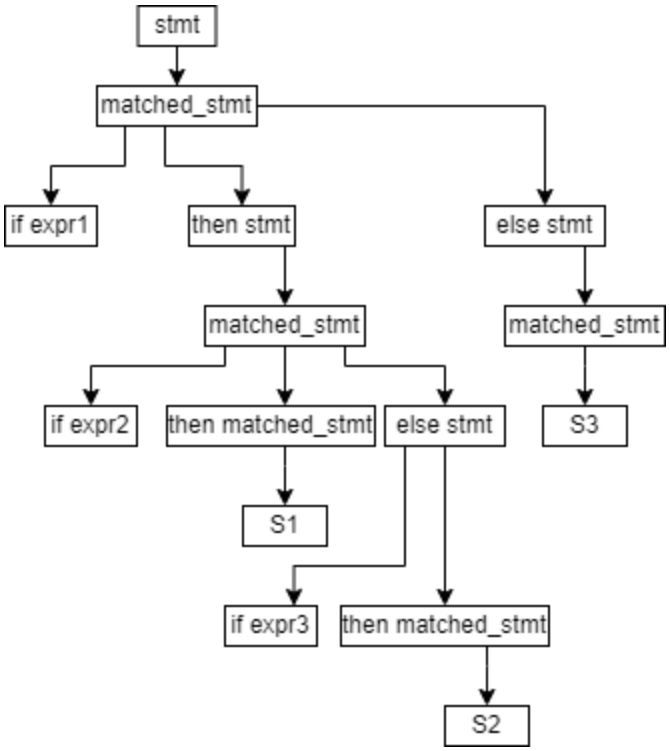
最右推导：$S \Rightarrow \underline{(L)} \Rightarrow (\underline{L,S}) \Rightarrow (L,\underline{(L)}) \Rightarrow (L,(\underline{L,S})) \Rightarrow (L,(L,\underline{(L)})) \Rightarrow (L,(L,(\underline{S}))) \Rightarrow (L,(L,(\underline{a}))) \Rightarrow (L,(\underline{S},(a))) \Rightarrow (L,(\underline{a},(a))) \Rightarrow (\underline{S},(a,(a))) \Rightarrow (\underline{a},(a,(a)))$



（2）习题3.5。

串if expr1 then if expr2 then S1 else if expr3 then S2 else S3，存在如下两种最左推导的分析树。





因此是二义的。

(3) 为以下文法构造LR(1)分析表：

(1) S→aSb

(2) S→A

(3) A→aA

(4) A→ε

FIRST(S)={a,ε}
FIRST(A)={a,ε}
FOLLOW(S)={b,$}
FOLLOW(A)={b,$}

状态	LR(1)项目	后继符号	后继状态
I0	[S'→·S,$]	S	I1
	[S→·A,$]	A	I2
	[S→·aSb,$]	a	I3
	[A→·aA,$]	a	I3
	[A→·,$]	$	
I1	[S'→S·,$]	$	
I2	[S→A·,$]	$	
I3	[S→a·Sb,$]	S	I4
	[S→·aSb,b]	a	I5
	[S→·A,b]	A	I6
	[A→·aA,b/$]	a	I5
	[A→·,b/$]	$	
	[A→a·A,$]	A	I6
I4	[S→aS·b,$]	b	I7
I5	[S→a·Sb,b]	S	I8
	[S→·aSb,b]	a	I5
	[S→·A,b]	A	I9
	[A→·aA,b/$]	a	I5
	[A→·,b/$]	$	
	[A→a·A,b/$]	A	I9
I6	[S→A·,b]	$	

状态	LR(1)项目	后继符号	后继状态
	[A→aA·,$]	$	
I7	[S→aSb·,$]	$	
I8	[S→aS·b,b]	b	I10
I9	[S→A·,b]	$	
	[A→aA·,b/$]	$	
I10	[S→aSb·,b]	$	

分析表如下：

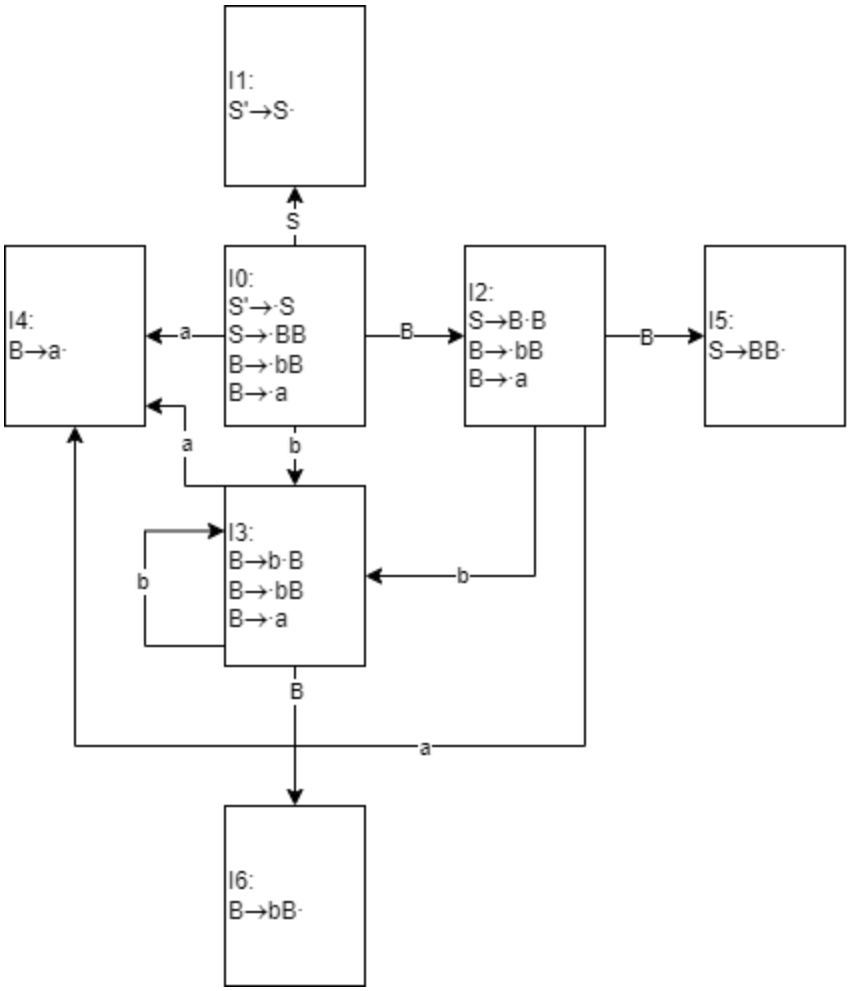
状态	Action			Goto	
	a	b	$	S	A
0	s3		r4	1	2
1			acc		
2			r2		
3	s5	r4	r4	4	6
4		s7			
5	s5	r4	r4	8	9
6		r2	r3		
7			r1		
8		s10			
9		r2/r3	r3		
10		r1			

（4）给出接受以下文法的活前缀且以LR(0)项目为状态的NFA,以及相应文法的SLR(1)分析表。

(1) S→BB
(2) B→bB
(3) B→a

FIRST(S)={a,b}
FIRST(B)={a,b}
FOLLOW(S)={ε}
FOLLOW(B)={a,b,ε}

状态	LR(0)项目	后继符号	后继状态
I0	S'→·S	S	I1
	S→·BB	B	I2
	B→·bB	b	I3
	B→·a	a	I4
I1	S'→S·	$	
I2	S→B·B	B	I5
	B→·bB	b	I3
	B→·a	a	I4
I3	B→b·B	B	I6
	B→·bB	b	I3
	B→·a	a	I4
I4	B→a·	$	
I5	S→BB·	$	
I6	B→bB·	$	



分析表如下：

状态	Action			Goto	
	a	b	$	S	B
0	s4	s3		1	2
1			acc		
2	s4	s3			5
3	s4				6
4	r3	r3			
5					
6	r2	r2			

（5）习题3.11，并描述该文法产生的语言，再给出该文法的递归下降分析程序。

(1) S→aBS|bAS|ε
(2) A→bAA|a
(3) B→aBB|b

FIRST(S)={a,b,ε}
FIRST(A)={a,b}
FIRST(B)={a,b}
FOLLOW(S)={$}
FOLLOW(A)={a,b,$}
FOLLOW(B)={a,b,$}

	a	b	$
S	S→aBS	S→bAS	S→ε
A	A→a	A→bAA	
B	B→aBB	B→b	

产生的语言为空串或a和b数量相等的任意ab的串。

```
void S(){
    if(lookhead=='a'){
        match('a');
        B();
        S();
    }
    else if(lookhead=='b'){
        match('b');
        A();
        S();
    }
    else return ;
}
void A(){
    if(lookhead=='a'){
        match('a');
    }
    else if(lookhead=='b'){
        match('b');
        A();
        A();
    }
    else error();
}
void B(){
    if(lookhead=='b'){
        match('b');
    }
    else if(lookhead=='a'){
        match('a');
        B();
        B();
    }
    else error();
}
```

(6) 习题3.21，3.24

3.21 (a)

```
(0) S→AaAb
(1) S→BbBa
(2) A→ε
(3) B→ε
```

FIRST(S)={a,b}
FIRST(A)={ε}
FIRST(B)={ε}
FOLLOW(S)={$}
FOLLOW(A)={a,b}
FOLLOW(B)={a,b}
FIRST(AaAb)={a},FIRST(BbBa)={b},这两个集合的交为空集，且都不包含ε，因此该文法是LL(1)文法。

状态	LR(0)项目	后继符号	后继状态
I0	S'→·S	S	I1
	S→·AaAb	A	I2
	S→·BbBa	B	I3
	A→·	$	
	B→·	$	
I1	S'→S·	$	
I2	S→A·aAb	a	I4
I3	S→B·bBa	b	I5
I4	S→Aa·Ab	A	I6
	A→·	$	
I5	S→Bb·Ba	B	I7
	B→·	$	
I6	S→AaA·b	b	I8
I7	S→BbB·a	a	I9
I8	S→AaAb·	$	

状态	LR(0)项目	后继符号	后继状态
I9	S→BbBa·	$	

SLR(1)分析表如下：

状态	Action			Goto		
	a	b	$	S	A	B
0	r2/r3	r2/r3			2	3

此处已经说明该文法不是SLR(1)文法，因此分析表不需要填写完整。

3.21 (b)

LL(1)文法中，非终结符A面临当前输入符号a时即可作出唯一的产生式的选择决定，也就是说，不会面临归约归约冲突，此外产生式不含左递归且具有相同左部的产生式不含左因子，相当于移进和归约的过程是不需要选择的，也就是不会面临冲突的。LR(1)也是从左往右看，并且只看下一个符号，再进行相应的动作。LR(1)比LL(1)拥有了更多的信息，在看到一个符号时，根据搜索符进行移进或规约动作。LL(1)文法的分析过程没有选择，那么根据搜索符的含义可知，LR(1)文法的分析过程能做到同样的分析，也就是LL(1)分析的每个步骤的移进或归约都有LR(1)分析中对应的一步。因此LL(1)文法是可以被LR(1)的分析过程正确分析的，LL(1)文法都是LR(1)文法。

3.24

```
(1) S→Aa
(2) S→bAc
(3) S→Bc
(4) S→bBa
(5) A→d
(6) B→d
```

FIRST(S)={b,d}
FIRST(A)={d}
FIRST(B)={d}
FOLLOW(S)={$}
FOLLOW(A)={a,c}
FOLLOW(B)={a,c}

状态	LR(1)项目	后继符号	后继状态
I0	[S'→·S,$]	S	I1
	[S→·Aa,$]	A	I2
	[S→·bAc,$]	b	I3
	[S→·Bc,$]	B	I4
	[S→·bBa,$]	b	I3
	[A→·d,a]	d	I5
	[B→·d,c]	d	I5
I1	[S'→S·,$]	$	
I2	[S→A·a,$]	a	I6
I3	[S→b·Ac,$]	A	I7
	[A→·d,c]	d	I8
	[S→b·Ba,$]	B	I9
	[B→·d,a]	d	I8
I4	[S→B·c,$]	c	I10
I5	[A→d·,a]	$	
	[B→d·,c]	$	
I6	[S→Aa·,$]	$	
I7	[S→bA·c,$]	c	I11
I8	[A→d·,c]	$	
	[B→d·,a]	$	
I9	[S→bB·a,$]	a	I12
I10	[S→Bc·,$]	$	
I11	[S→bAc·,$]	$	
I12	[S→bBa·,$]	$	

LR(1)分析表如下：

状态	Action					Goto		
	a	b	c	d	$	S	A	B
0		s3		s5		1	2	4
1					acc			
2	s6							
3				s8			7	9
4			s10					
5	r5		r6					
6					r1			
7			s11					
8	r6		r5					
9	s12							
10					r3			
11					r2			
12					r4			

分析表中未出现冲突，因此是LR(1)文法。
注意到I5和I8是同心集，合并后得到I58：

状态	项目	后继符号	后继状态
I58	[A→d·,a/c]	$	
	[B→d·,a/c]	$	

显然I58在计算分析集的过程中，会出现归约归约冲突。因此该文法不是LALR(1)文法。

（7）指出其中语句1所对应的汇编代码；给语句2中表达式添加嵌套小圆括号()来表示计算次序，越内层越优先计算;再给出每层()所对应的主要汇编代码。例如，(p)将是最内层括号。

```
int *p = &i;//语句1
```

对应的汇编代码为

```
addi     a5,s0,-24
sw      a5,-20(s0)
```

语句2

```
++(*(p++));
```

以下分别给出p++,*(),++()对应的汇编代码

```
lw      a5, -20(s0)  # 加载p的地址到 a5
addi    a4, a5, 4    # 计算p++，即将p指向下一个地址
sw      a4, -20(s0)  # 更新p的值到栈中


lw      a4, 0(a5)    # 取出p++之前的地址所指向的值*(p++)


addi    a4, a4, 1    # 对上一步取出的值加一++(*(p++))
sw      a4, 0(a5)    # 将自增后的值存回到原地址中
```