H7−1, H7−2

为文法:

$$S \rightarrow (L) \mid a$$
$$L \rightarrow L, S \mid S$$

(1)写一个翻译方案，它打印出每个a在句子中是第几个字符。例如，当句子是(a,(a,(a,a),(a)))时，打印的结果是2， 5， 8， 10，14。

(2)写出相应的语法制导定义

(3)写出相应的预测翻译器

(4)写出自下而上分析的栈操作代码

概念区分

- 语义规则和产生式相联系的两种方式
  - 语法制导定义
    - 将文法符号和某些属性相关联，并通过语义规则来描述如何计算属性的值，没有描述这些规则的计算时机
  - 语法制导的翻译方案
    - 在产生式的右部的适当位置，插入相应的语义动作，按照分析的进程，执行遇到的语义动作，从而明确了语法分析过程中属性的计算时机。

- a自身的信息无法确定a在序列中的位置，因此必须要借助继承属性。
- 方法一：
  - 继承属性 in：该文法符号推出的字符序列的前面已经有多少字符
  - 综合属性 out：该文法符号推出的字符序列的最后一个字符在序列中是第几个字符

- 4.12(b) 文法如下：

  $S \rightarrow (L) \mid a$

  $L \rightarrow L,S \mid S$

(1)写一个翻译方案，它打印出每个a在句子中是第几个字符。例如，当句子是(a,(a,(a,a),(a)))时，打印的结果是2， 5， 8， 10， 14。

```
S’  →  { S.in = 0; }  S
S   →  { L.in = S.in +1; }  (L)  { S.out = L.out + 1; }
S   →  a { S.out = S.in + 1; print (S.out); }
L   →  { L1.in = L.in; }  L1, { S.in = L1.out + 1; } S {L.out = S.out; }
L   →  { S.in = L.in; }  S  { L.out = S.out; }
```

- a自身的信息无法确定a在序列中的位置，因此必须要借助继承属性。
- 方法二：
  - 继承属性 in：该文法符号推出的字符序列的前面已经有多少字符
  - 综合属性 total：该文法符号推出的字符序列所包含的字符总数

- 4.12(b) 文法如下：

  $S \rightarrow (L) \mid a$

  $L \rightarrow L,S \mid S$

(1)写一个翻译方案，它打印出每个a在句子中是第几个字符。例如，当句子是(a,(a,(a,a),(a)))时，打印的结果是2， 5， 8， 10， 14。

```
S’  →  { S.in = 0; }  S
S   →  { L.in = S.in +1; }  (L)  { S.total = L.total + 2; }
S   →  a { S.total = 1; print (S.in + 1); }
L   →  { L1.in = L.in; }  L1, { S.in = L1.in + L1.total + 1; }
S {L.total = L1.total +S.total + 1; }
L   →  { S.in = L.in; }  S  { L.total = S.total; }
```

- a自身的信息无法确定a在序列中的位置，因此必须要借助继承属性。
- 方法一的语法制导定义：
  - 继承属性 in：该文法符号推出的字符序列的前面已经有多少字符
  - 综合属性 out：该文法符号推出的字符序列的最后一个字符在序列中是第几个字符

- 4.12(b) 文法如下：

  $S \rightarrow (L) \mid a$

  $L \rightarrow L,S \mid S$

写出相应的语法制导定义

产生式	语义规则
S’ → S	S.in = 0;
S → (L)	L.in = S.in +1; S.out = L.out + 1;
S → a	S.out = S.in + 1; print (S.out);
L → L1, S	L1.in = L.in; S.in = L1.out + 1; L.out = S.out;
L → S	S.in = L.in; L.out = S.out;

- 消除左递归

S’ → S
S → (L)
S → a
L → ST
T → ,ST | ε

- 4.12(b) 文法如下：

  $S \rightarrow (L) \mid a$

  $L \rightarrow L,S \mid S$

写出相应的预测翻译器

产生式	语义规则
S’ → S	S.in = 0;
S → (L)	L.in = S.in +1; S.out = L.out + 1;
S → a	S.out = S.in + 1; print (S.out);
L → ST	S.in = L.in; T.in = S.out; L.out = T.out;
T → ,ST_1	S.in = T.in + 1; T_1.in = S.out; T.out = T_1.out
T → ε	T.in = T.out

产生式	语义规则
S’ → S	S.in = 0;
S → (L)	L.in = S.in +1; S.out = L.out + 1;
S → a	S.out = S.in + 1; print (S.out);
L → ST	S.in = L.in; T.in = S.out; L.out = T.out;
T → ,ST_1	S.in = T.in + 1; T_1.in = S.out; T.out = T_1.out
T → ε	T.in = T.out

```
int S’(){
  return S(0);
}
```

```
int S(int b){
  int in, out;
  if(lookahead == ‘(’){
      in = b + 1;
      match(‘(’);
      out = L(in) + 1;
      match(‘)’)
  }else
  {
       match(‘a’);
       out = b + 1;
       print(out);
  }
   return out;
}
```

产生式	语义规则
S’ → S	S.in = 0;
S → (L)	L.in = S.in +1; S.out = L.out + 1;
S → a	S.out = S.in + 1; print (S.out);
L → ST	S.in = L.in; T.in = S.out; L.out = T.out;
T → ,ST_1	S.in = T.in + 1; T_1.in = S.out; T.out = T_1.out
T → ε	T.in = T.out

```
int L(int b){
  int out;
  out = S(b);
  out = T(out);
  return out;
}
```

```
int T(int b)
{
   int out;
   if(lookahead == ‘,’){
      match(‘,’);
      out = S(b+1);
      out = T(out);
    }else{
      out = b;
    }
    return out;
}
```

- 引入标记非终极符M, N, R, P

- 4.12(b) 文法如下：

S’ → S　　S → (L)
S → a
L → ST　　T → ,ST | ε

写出自下而上分析的栈操作代码

产生式	语义规则	栈操作代码
S’ → MS	S.in = M.out	Stack[top - 1] = Stack[top]
M → ε	M.out = 0	Stack[top + 1] = 0
S → (NL)	N.in = S.in + 1, L.in = N.out; S.out = L.out + 1;	Stack[top - 3] = Stack[top - 1] + 1
N → ε	N.out = N.in	Stack[top + 1] = Stack[top - 1] + 1
S → a	S.out = S.in + 1; print (S.out);	Stack[top] = Stack[top - 1] + 1
L → SRT	S.in = L.in; R.in = S.in; T.in = R.out, L.out = T.out;	Stack[top - 2] = Stack[top]
R → ε	R.out = R.in	Stack[top + 1] = Stack[top - 1]
T → ,SPT$_1$	S.in = T.in + 1; P.in = S.in; T$_1$.in = P.out; T$_1$.out = S.out;	Stack[top - 3] = Stack[top]
P → ε	P.out = P.in	Stack[top + 1] = Stack[top]
T → ε	T.out = T.in	Stack[top] = Stack[top - 1]

期中考试第三题

3、（7 分）针对第 2 题的**语言**（注意这里文法可以是自己写的非二义文法），

1）写一语法制导的翻译方案，它针对输入的每个合法的串，以二元组形式依次输出该串的每个最长匹配的“a 比 b 多一个且不含 a 和 b 个数相等的非空后缀”（当串为 a 的个数比 b 的个数多）或“b 比 a 多一个且不含 a 和 b 个数相等的非空后缀”（当串为 a 的个数比 b 的个数少）的子串在串中的位置，再换行输出多几个 a 或 b。

例如，对于串 aba，会输出：

(1, 13)

1a

对于串 $aabaaaba$，会先后依次输出：

(1,1) (2,24) (35,5) (6,68)

4 a

2）将语法制导的翻译方案变成栈操作代码。

非二义文法

$S \to A \mid B$　　$A \to CA \mid CE$　　$B \to DB \mid DE$

$C \to a \mid bCC$　　$D \to b \mid aDD$　　$E \to aDE \mid bCE \mid \varepsilon$

E 表示 a 和 b 的个数相等的串；

C 表示 a 比 b 多一个的子串，并且推出的串中不含 a 和 b 个数相等的非空后缀

D 表示 b 比 a 多一个的子串，并且推出的串中不含 a 和 b 个数相等的非空后缀

翻译方案

3、A.i、A.n 表示 A 对应的终结符串的第 1 个符号的位置和 A 中 a 的个数比 b 的多几个

B.i、B.n 表示 B 对应的终结符串的第 1 个符号的位置和 B 中 b 的个数比 a 的多几个

C.i、C.s 表示 C 对应的终结符串的第 1 个符号的位置和最后 1 个符号的位置

D.i、D.s 表示 D 对应的终结符串的第 1 个符号的位置和最后 1 个符号的位置

E.i、E.s 表示 E 对应的终结符串的第 1 个符号的位置和最后 1 个符号的位置

$S \rightarrow \{ A.i = 1; \} A \{ print(\text{“\textbackslash n”}+A.n+\text{“ a”}); \}$

$S \rightarrow \{ B.i = 1; \} B \{ print(\text{“\textbackslash n”}+B.n+\text{“ b”}); \}$

$A \rightarrow \{ C.i = A.i; \} C \{ A_1.i = C.s+1; print(\text{“(”}+A.i+\text{“,”}+C.s+\text{“)”}); \} A_1 \{ A.n = A_1.n+1; \}$

$A \rightarrow \{ C.i = A.i; \} C \{ E.i = C.s+1; print(\text{“(”}+A.i+\text{“,”}+C.s+\text{“)”}); \} E \{ A.n = 1; \}$

$B \rightarrow \{ D.i = B.i; \} D \{ B_1.i = D.s+1; print(\text{“(”}+B.i+\text{“,”}+D.s+\text{“)”}); \} B_1 \{ B.n = B_1.n+1; \}$

$B \rightarrow \{ D.i = B.i; \} D \{ E.i = D.s+1; print(\text{“(”}+B.i+\text{“,”}+D.s+\text{“)”}); \} E \{ B.n = 1; \}$

$C \rightarrow a \{ C.s = C.i; \}$

$C \rightarrow b \{ C_1.i = C.i+1; \} C_1 \{ C_2.i = C_1.s+1; \} C_2 \{ C.s = C_2.s; \}$

$D \rightarrow b \{ D.s = D.i; \}$

$D \rightarrow a \{ D_1.i = D.i+1; \} D_1 \{ D_2.i = D_1.s+1; \} D_2 \{ D.s = D_2.s; \}$

$E \rightarrow a \{ D.i = E.i+1; \} D \{ E_1.i = D.s+1; \} E_1 \{ E.s = E_1.s; \}$

$E \rightarrow b \{ C.i = E.i+1; \} C \{ E_1.i = C.s+1; \} E_1 \{ E.s = E_1.s; \}$

$E \rightarrow \{ E.s = E.i - 1; \}$

栈操作代码

```
S → L A { print("\n"+val[top]+" a"); }
L → ε { val[top+1] = 1; }
S → L B { print("\n"+ val[top]+" b"); }
A → C M A1 { val[top-2] = val[top]+1; }
M → ε { val[top+1] = val[top]+1; print("("+val[top-1] +","+ val[top] +")"); }
A → C M E { val[top-2] = 1; }
B → D N B1{ val[top-2] = val[top]+1; }
N → ε { val[top+1] = val[top]+1; print("("+val[top-1] +","+ val[top] +")"); }
B → D N E { val[top-2] = 1; }
C→ a { val[top] = val[top-1]; }
C→ b P C1 Q C2 { val[top-4] = val[top]; }
P → ε { val[top+1] = val[top-1]+1; }
Q → ε { val[top+1] = val[top]+1; }
D→ b { val[top] = val[top-1]; }
D→a P D1 Q D2 { val[top-4] = val[top]; }
E→ a P D Q E1{ val[top-4] = val[top];}
E→ b P C Q E1{ val[top-4] = val[top];}
E→ { val[top+1] = val[top]; }
```