编译原理典型案例

1. 对于文法 G[S]

S →(L)

S→aS

S→a

L →L,S

L→S

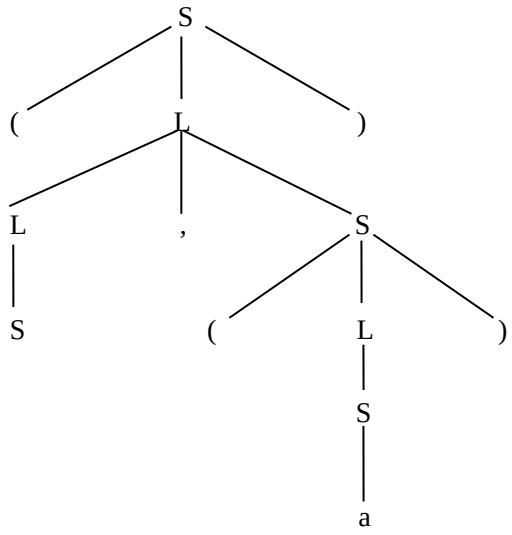
(1) 画出句型 (S,(a)) 的语法树；

(2) 写出上述句型的所有短语、直接短语、句柄和素短语。

解答

这类题目重点考查语法树、推导、短语、直接短语、句柄和素短语等基本概念。在句型中寻找短语、直接短语、句柄的方法：

(1) 画出句型对应的语法树。句型 (S,(a)) 的语法树如下图所示



(2) 在该语法树中寻找短语、直接短语、句柄。首先我们看短语的定义：令 G 是一个文法，S 是文法的开始符，假设 α , β , δ 是文法 G 的句型，如果有

$$S^{*}\Rightarrow \alpha A \delta \quad 且 \quad A^{+}\Rightarrow \beta$$

则称 β 是句型 α β δ 相对于非终结符 A 的短语。如果有 $A\Rightarrow \beta$，则称 β 是句型 α β δ 相对于规则 A→ β 的直接短语。一个句型的最左直接短语称为该句型的句柄。根据短语的定义可知，以非终结符 A 为根的子树的叶结点从左到右排列就是相对于非终结符 A 的短语；如果子树只有两代，则短语就是直接短语；最左边的两代子树的所有叶结点从左到右排列，就是该句型的句柄。素短语是一个短语，它至少包含一个终结符，且除自身外不再包含其他素短语。处于句型最左边的素短语为最左素短语。

从语法树中我们可以找到：

短语：a，S，(a)， S, (a)， (S, (a))

直接短语：a，S

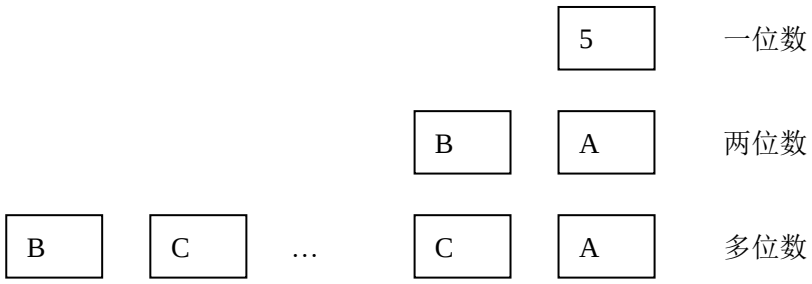
句柄：S

素短语：a

2. 写一个上下文无关文法，使其语言能被 5 整除且不以 0 开头的无符号整数集合（如{5，10，15}）

解答

能被 5 整除的数从形式上看，是以 0，5 结尾的数字串。题目要求不以 0 开头，注意 0 不是该语言的句子。句子的结构分为三种：



其中，A 代表可以出现在个位上的数字，只能是 0 或 5；

B 代表可以出现在开始位上的数字，除了 0 以外，其他数字都可以；

C 代表可以出现在中间位上的数字。0-9 所有数字都可以。

于是，A→0 | 5

B→1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

C→0 | B

写文法时，先描述一位数结构，于是有产生式 S →5。对于两位数和多位数，都是以 B 开头和以 A 结尾，于是有产生式 S→DA。用非终结符 D 推导出两位数和多位数中除个位数字以外的数字。对与多位数，由于中间位可以是许多位，必须使用递归技术来描述其结构。于是有产生式：

D→DC

D→B

因此，所求文法为 G[S]：

S →5

S→DA

D→DC

D→B

A→0 | 5

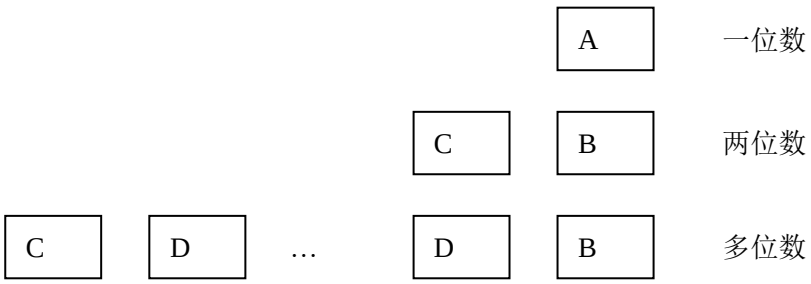
B→1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

C→0 | B

3. 写一个文法 G, 使其语言为 不以 0 开头的偶数集。

解答

不以 0 开头的偶数集数字的结构分为三种：一位数、两位数和多位数。



其中，A 代表可以出现在个位上的数字，可以是 2，4，6，8，但不能是 0；

B 代表可以出现在开始位上的数字，除了 0 以外，其他数字都可以；

C 代表可以出现在中间位上的数字。0-9 所有数字都可以。

于是，A→2 | 4 | 6 | 8

B→0 | A

C→1 | 3 | 5 | 7 | 9 | A

D→0 | C

写文法时，先描述一位数结构，于是有产生式 S →A。对于两位数和多位数，都是以 C 开头和以 B 结尾，于是有产生式 S→CE。用非终结符 E 推导出两位数和多位数中除个位数字以外的数字。对与多位数，由于中间位可以是许多位，必须使用递归技术来描述其结构。于是有产生式：

E→CE

E→B

因此，所求文法为 G(S)：

S→A | CE

E→CE | B

A→2 | 4 | 6 | 8

B→0 | A

C→1 | 3 | 5 | 7 | 9 | A

D→0 | C

4. 考虑下面的程序：

```
…
procedure P(x, y, z);
    begin
      y:=y+1;
      z:=z+x
    end;
    begin
      a:=2;
      b:=3;
```

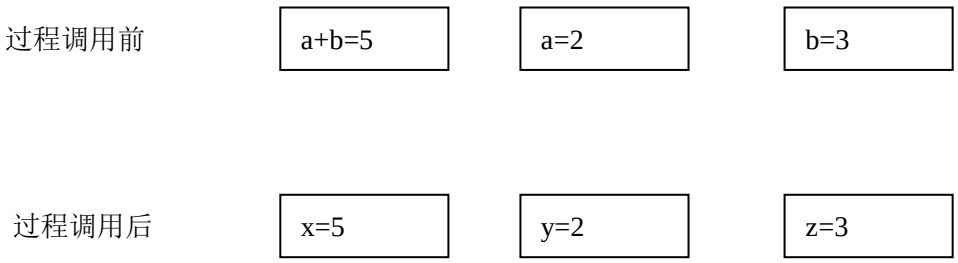
```
    P(a+b, a, a);
    print a
end.
```

试问，若参数传递的方式分别采用传值、传地址、得结果和传名时，程序执行后输出 a 的值是什么?

解答

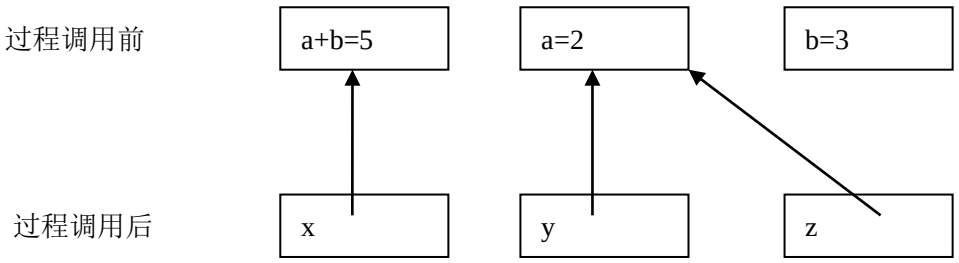
所谓传值是调用段把实在参数的值计算出来，被调用段把这些值抄进自己的形式参数中，像使用局部名一样使用这些形式单元。对形式参数的任何运算不影响实参的值。

上面过程 P 调用的的参数传递过程如下图所示。



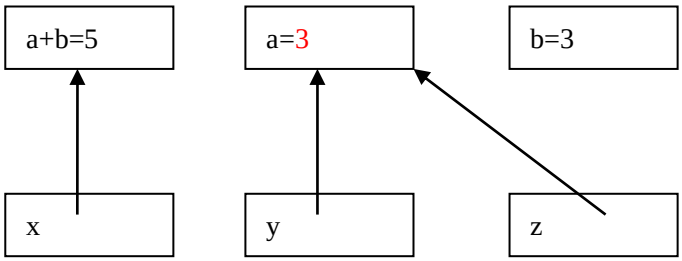
但过程 P 无法改变实参 a 的值。因此，程序执行后输出 a 的值是 2。

所谓传地址是把实在参数的地址传递给相应的形式参数。在过程段中每个形式参数都有一个相应的单元，称为形式单元。形式单元将用来存放相应实在参数的地址。过程体对形式参数的任何引用或赋值都被处理成对形式单元的间接访问。

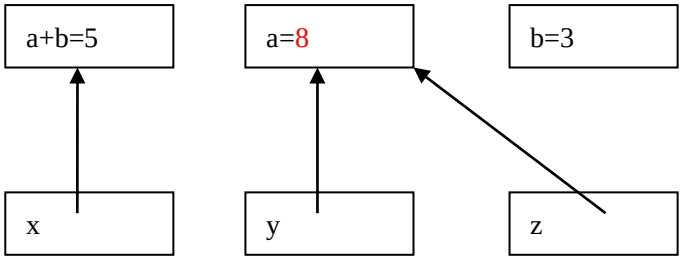


过程调用后，形参 x 的形式单元指向存 a+b 值的临时变量的地址，形参 y 的形式单元指向变量 a 的地址，形参 z 的形式单元指向变量 b 的地址。形参通过指针可以间接访问实参。

执行 y:=y+1; 后，实在参数的变化：



执行 z:=z+x; 后，实参的变化：

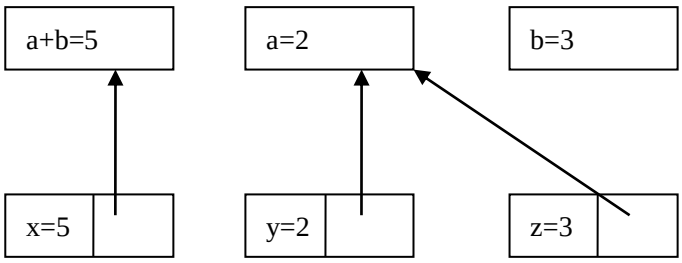


因此，程序执行后输出 a 的值是 8。

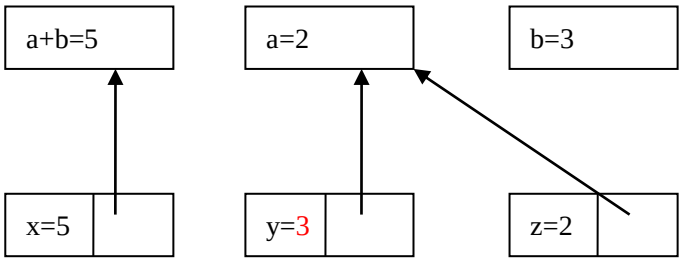
所谓得结果是每个形式参数对应两个单元，第一个单元存放实参的地址，第二个单元存放实参的值。在过程体中对形参的任何引用或赋值都被处理成对第二个形式单元的直接访问，但在过程工作完成返回之前必须把第二个单元的内容存放到第一个单元所指的那个实参单元之中。

过程调用前

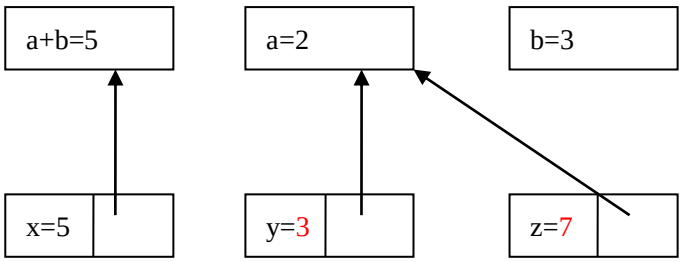
过程调用后



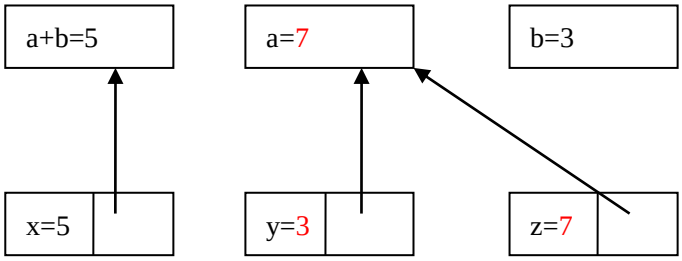
执行 y:=y+1; 后，实参和的形参的变化：



执行 z:=z+x; 后，实参和的形参的变化：



过程工作完成返回之前必须把第二个单元的内容存放到第一个单元所指的那个实参单元之中。实参和的形参的变化：



因此，程序执行后输出 a 的值是 7。

(4) 所谓传名是在进入调用段之前不对实在参数预先进行计值，而是过程中每当使用到相应的形参时才对它实行计值。因此,在实现时通常都把实参处理成型个子程(称为参数子程序),每当过程体中使用到相应形参时就调用这个子程序。

因此，过程体执行 y:=y+1;语句，实现时处理成为：

a=a+1;

过程体执行 z:=z+x;语句，实现时处理成为：

a=a+(a+b);

执行上述两语句后，a 的值是 9。因此，程序执行后输出 a 的值是 9。

综上所述程序执行时 a 的输出：

(1)传值： 2

(2)传地址： 8

(3)得结果： 7

(4)传名： 9

5. 已知文法 G[S]

S→S*aP | aP | *aP

P→+aP | +a

(1) 将文法 G[S]改写为 LL(1)文法 G'[S]；

(2) 构造文法 G'[S]的预测分析表。

解答

构造文法的预测分析表，通常应当按下列步骤进行：

(1) 消除文法的左递归

(2) 对消除左递归后的文法，提取左公因子。

(3) 对经过上述改造后的文法，计算它的每个非终结符的 FIRST 和 FOLLOW 集合。

(4) 根据 FIRST 和 FOLLOW 集合构造预测分析表。

消除直接左递归方法：假设关于非终结符 P 的规则为

P→P α | β

其中，β 不以 P 打头。把 P 的规则改写为如下非直接左递归形式：

P→ β P '

P→ α P '|ε

文法 G[S] 消除直接左递归方法后，文法变为 G'[S]：

S→aPS' | *aPS'

S'→*aPS' | ε

P→+aP | +a

提公共左因子的方法：假设关于非终结符 A 的规则为

A→ δ β 1| δ β 2|…| δ β n| γ 1| γ 2|…| γ m

其中，每个 γ 不以 δ 开头。把 A 的规则改写为如下形式：

A→ α A'| γ 1| γ 2|…| γ m

A'→ β 1| β 2|…| β n

于是，文法 G'(S) 提公共左因子后，文法变为 G"[S]

S→aPS' | *aPS'

S’→*aPS’ | ε

P→+aP’

P’→P | ε

计算每个非终结符的 FIRST 和 FOLLOW 集合：

a. 计算非终结符 X 的 FIRST 方法：

(1)若有产生式 X→a…，则把 a 加入到 FIRST(X)中；

(2)若 X→ε 也是一条产生式,则把 ε 也加到 FIRST(X)中；

(3)若 X→Y…是一个产生式且 Y∈VN，则把 FIRST(Y)中的所有非 ε 元素都加到 FIRST(X)中；若 X→Y1Y2…YK 是一个产生式，Y1,Y2,…,Yi-1 都是非终结符，而且，对于任何 j，1≤j≤i-1,FIRST(Yj)都含有 ε，则把 FIRST(Yj)中的所有非 ε 元素都加到 FIRST(X)中；特别是，若所有的 FIRST(Yj)(j=1,2,…,K)均含有 ε，则把 ε 加到 FRIST(X)中。

根据计算非终结符的 FIRST 方法的第一点，有

FIRST(S)={a，*}

FIRST(S’)={*}

FIRST(P)={+}

FIRST(P’)={}

根据计算非终结符的 FIRST 方法的第二点，有

FIRST(S)={a，*}

FIRST(S’)={*，ε}

FIRST(P)={+}

FIRST(P’)={ ε}

根据计算非终结符的 FIRST 方法的第三点，有

FIRST(S)={a，*}

FIRST(S’)={*，ε}

FIRST(P)={+}

FIRST(P’)={+，ε}

b. 计算非终结符 X 的 FOLLOW 方法：

(1)对于文法的开始符号 S,置#于 FOLLOW(S) 中；

(2).若 A→αBβ 是一个产生式，则把 FIRST(β)\{ε}加至 FOLLOW(B)中；

(3)若 A→αB 是一个产生式,或 A→αBβ 是一个产生式而 β⇒ε(即 ε∈FIRST(β))，则把 FOLLOW(A)，加至 FOLLOW(B)中。

根据计算非终结符的 FOLLOW 方法的第一点，有

FOLLOW(S)={#}

FOLLOW(S’)={}

FOLLOW(P)={}

FOLLOW(P’)={}

根据计算非终结符的 FOLLOW 方法的第二点，有

FOLLOW(S)={#}

FOLLOW(S’)={#}

FOLLOW(P)={*，#}

FOLLOW(P’)={*，#}

根据 FIRST 和 FOLLOW 集合构造预测分析表方法：

(1)对文法 G 的每个产生式 A→α 执行第二步和第三步；

(2)对每个终结符 a∈FIRST(α)，把 A→α 加至 M[A,a]中；

(3)若 ε∈ FIRST(α)，则对任何 b∈FOLLOW(A)，把 A→α 加至 M[A,b]中；

(4)把所有无定义的 M[A,a]标上“出错标志”。

构造预测分析表方法如下：

	*	+	a	#
S	S→*aPS’		S→aPS’	
S’	S’→*aPS’			S’→ε
P		P→+aP’		
P’	P’→ε	P’→P		P’→ε

6. 设文法 G(S):

S→(T) | aS | a

T→T,S | S

⑴ 消除左递归和提公共左因子；

⑵ 构造相应的 FIRST 和 FOLLOW 集合；

⑶ 构造预测分析表。

解答

(1)消除左递归和提公共左因子，文法变为 G’[S]

S →(L) | aS’

S’→S |ε

L→SL’

L’→,SL’ |ε

此文法没有公共左因子

(2) 构造相应的 FIRST 和 FOLLOW 集合

FIRST(S)={a，(}

FIRST(S’)={a，(，ε}

FIRST(L)={a, (}

FIRST(L’)={,, ε}

FOLLOW(S)={,,), #}

FOLLOW(S’)={,,), #}

FOLLOW(L)={)}

FOLLOW(L')={)}

(3) 构造预测分析表

	(	)	a	,	#
S	S →(L)		S → aS'		
S'	S'→S	S'→ ε	S'→S	S'→ ε	S'→ ε
L	L→SL'		L→SL'	L'→,SL'	
L'		L'→ ε			

7. 设文法 G[S]

S →(A)

S →a

A →A+S

A →S

(1) 构造每个非终结符的 FIRSTVT 和 LASTVT 集合；

(2) 构造优先关系表；

解答

对于这类题目，关键是准确掌握 FIRSTVT 和 LASTVT 集合的定义和构造方法。

FIRSTVT(P)={a|P^{+}⇒a…或 P^{+}⇒Qa...，a∈VT 而 Q∈VN}

即，对于非终结符 P，其往下推导所可能出现的首个算符。

LASTVT(P)={a|P^{+}⇒…a 或 P^{+}⇒...aQ，a∈VT 而 Q∈VN}

即，对于非终结符 P，其往下推导所可能出现的最后一个算符。

构造 FIRSTVT 和 LASTVT 集合的方法

(1) 构造 FIRSTV 集合

若有产生式 P→a…或 P→Qa…，则 a∈FIRSTVT(P)；

若有 a∈FIRSTVT(Q)，且有产生式 P→Q…，则 a∈FIRSTVT(P)。

(2) 构造 LASTVT 集合

若有产生式 P→…a 或 P→…aQ，则 a∈LASTVT(P);

若有 a∈LASTVT(Q)，且有产生式 P→…Q，则 a∈LASTVT(P)。

对于文法 G[S]，构造它的每个非终结符的 FIRSTVT 和 LASTVT 集合：

FIRSTVT(S)={a, (}

FIRSTVT(A)={+, a, (}

LASTVT(S)={a,)}

LASTVT(A)={+, a,)}

构造算符优先关系表方法：

(1) '='关系

直接看产生式的右部，若出现了

P →…ab…或 P →…aQb，则 a=b

(2)'<'关系

求出每个非终结符 P 的 FIRSTVT(P)

若 P→…aR…，则∀b∈FIRSTVT(R)，a<b

(3)'>'关系

求出每个非终结符 P 的 LASTVT(P)

若 P→…Rb…，则∀a∈LASTVT(B)，a>b

对于文法 G[S]，构造它的算符优先关系表如下：

	a	+	(	)
a		>		>
+	<	>	<	>
(	<	<	<	=
)		>		>

8. 设文法 G（S）：

S→T | S∨T

T→U |T∧U

U→i |-U

(1) 计算 FIRSTVT 和 LASTVT；

(2) 构造优先关系表。

解答

(1)对于文法 G[S]，构造它的每个非终结符的 FIRSTVT 和 LASTVT 集合。

FIRSTVT(S)={∨, ∧, i, - }

FIRSTVT(T)={∧, i, -}

FIRSTVT(U)={i, -}

LASTVT(S)={∨, ∧, i, - }

LASTVT(T)={∧, i, -}

LASTVT(U)={i, -}

(2) 构造优先关系表。

	i	∨	∧	-
S		.>	.>	
∨	<.	.>	<.	<.
∧	<.	.>	.>	<.
-	<.	.>	.>	<.

9. 下面映射 if 语句的文法 G[S]是算符优先文法吗？如果是，则构造算符优先关系表。如果不是，则说明理由。

G[S]

S→iBtS

S→iBtSeS

S→a

B→b

解答

对于文法 G[S]，它是一个二义性文法。因为存在句子 ibtibtaea 有两个不同的最左推导

S⇒ iBtS⇒ ibtS⇒ ibtiBtSeS⇒ ibtibtSeS⇒ ibtibtaeS⇒ ibtibtaea

S⇒ iBtSeS⇒ ibtSeS⇒ ibt iBtSeS⇒ ibtibtSeS⇒ ibtibtaeS⇒ ibtibtaea

由于算符优先文法一定不是二义性文法，所以文法 G[S]不是算符优先文法。

10. 设有如下的基本块

T1:=A+B

T2:=5

M:=T2*4

T3:=C-D

T4:=M+T3

L:=T1*T3

T4:=A+B

N:=T4

(1) 画出该基本块的 DAG 图；

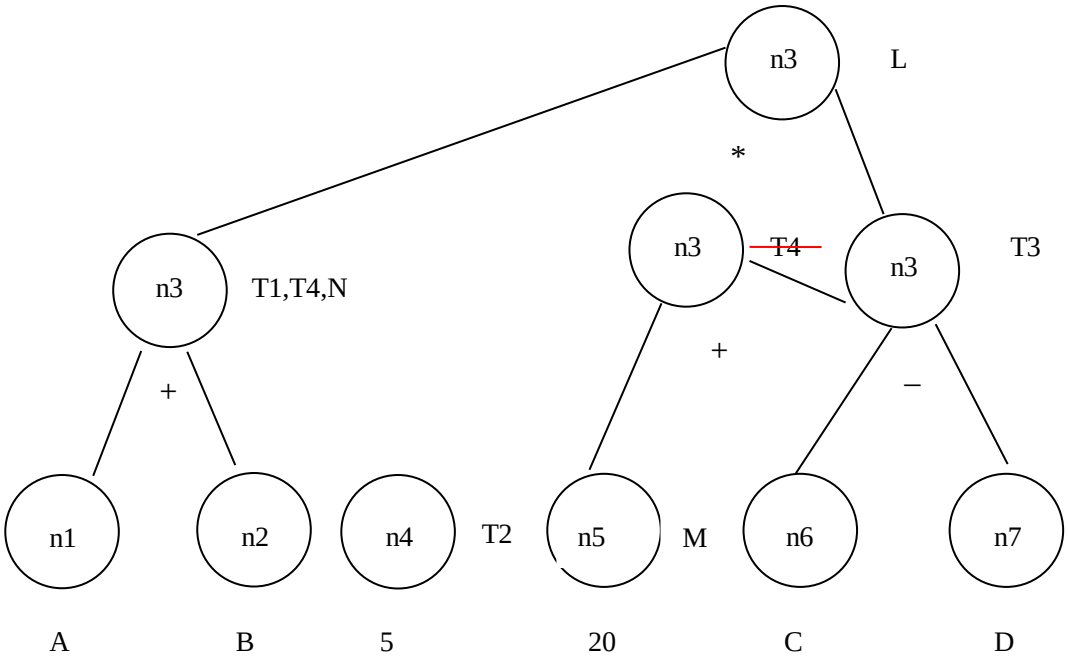
(2) 假设只有 L，M 和 N 在基本块之后还要引用，写出优化后的四元式序列。

解答

在构造基本块的 DAG 图时，必须注意两个方面：一是对于已知的常量要进行计算；二是对同一变量进行多次定值时，最后一次定值是基本块中该变量的最终变量。由于基本块有语句 T2：=5，可知 T2 是已知量。所以，语句 M：=T2 * 4 可计算出 M 的值来，即，M 也转为常量（M = 20）。一方面，变量 T4 进行了两次操作，基本块执行完时，最后一次给 T4 的定才是 T4 的最终结果。活跃信息是针对变量而言的。如果一个变量在基本块之后不再被引用，则称该变量是非活跃的（或称为非活跃变量），反之称之为活跃的（或称为活跃变量）。变量的活跃信息对代码优化和代码生成都有非常重要的指导意义。

当 DAG 图构造出来后，根据题目所给的条件从 DAG 图中计算出活跃变量的结果即可。本是中只要计算出变量 L、M 和 N 的结果就行了。

基本块对应的 DAG 图如下：



按照构造其结点的顺序，重新写成四元式序列 G’

T1=A+B
T4=T1
N=T4
T2=5
M=20
T3=C-D
L=T1*T3

由于只有 L，M 和 N 在基本块之后还要引用，T1、T2、T3 和 T4 都不会引用，于是重新写出优化后的四元式序列为：

N:=A+B
M:=20
T3:=C-D
L:=N*T3

11. 把语句

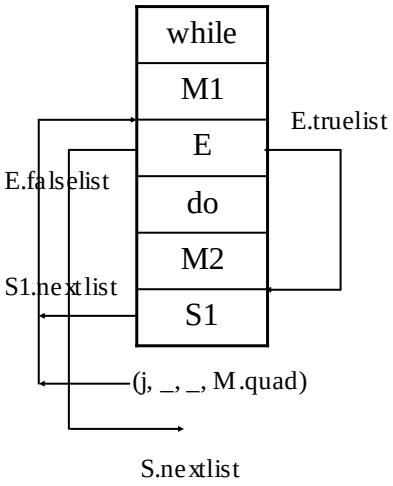
while a>0 ∧b>0 do
if x>y then b:=b-1
else a:=a-1;

翻译为四元式序列。

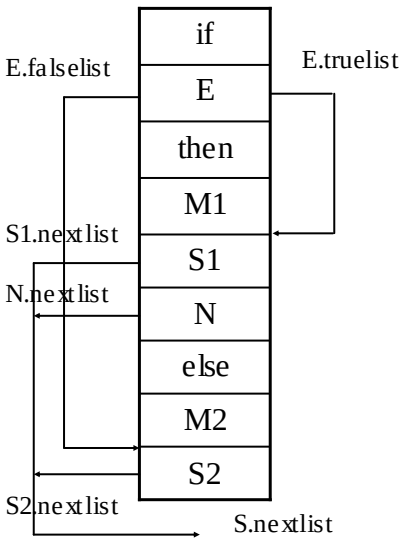
解答

对于这类题目，关键是准确掌握语句的语动作。

while-do 语句的语义动作



if-then-else 语句的语义动作



作为条件控制的布尔式的翻译：

把 A or B 解释成　　if A then true else B

把 A and B 解释成　　if A then B else false

把 not A 解释成　　if A then false else true

布尔表达式赋予两种“出口”：一是“真”出口；一是“假”出。

假设四元式序列从 100 开始编号。

表达式 a>0 翻译为：

100 (j>，a，0，_)

101 (j，_，_，_)

继续翻译表达式 a>0 ∧b>0，由于是与运算，100 号语句应转跳到 102 号语句，102 号语句为“真”出口，101 号语句和 103 号语句为“假”出，四元式序列为：

100 (j>，a，0，102)

101 (j，_，_，_)

102 (j>，b，0，_)

103 (j，_，_，_)

继续翻译语句 while a>0 ∧b>0 do

if x>y …

while 语句中的逻辑表达式的“真”出口应转跳到其循环体的第一个四元式标号，即 104 号语句。

100 (j>，a，0，102)

101 (j，_，_，_)

102 (j>，b，0，104)

103 (j，_，_，_)

104 (j>，x，y_)

105 (j，_，_，_)

if 语句中的逻辑表达式的“真”出口为 104 号语句，应转跳到 then 中的第一个四元式标号，“假”出口为 105 号语句，应转跳到 else 中的第一个四元式标号。then 中语句翻译完，应转跳出去。

100 (j>，a，0，102)

101 (j，_，_，_)

102 (j>，b，0，104)

103 (j，_，_，_)

104 (j>，x，y_)

105 (j，_，_，_)

106 (-，b，1，T1)

107 (:=，T1，_，b)

108 (j，_，_，_)

109(-，a，1，T2)

110 (:=，T2，_，a)

while 语句中循环体翻译完，应转跳到 while 语句的第一个四元式标号，即 100 号语句。while 语句中的逻辑表达式的“假”出口应转跳到 while 语句后的第一个四元式标号。

100 (j>，a，0，102)

101 (j，_，_，112)

102 (j>，b，0，104)

103 (j，_，_，112)

104 (j>，x，y_)

105 (j，_，_，_)

106 (-，b，1，T1)

107 (:=，T1，_，b)

108 (j，_，_，100)

109(-，a，1，T2)
110 (:=，T2，_，a)
111 (j，_，_，100)

12. 已知文法 G[S]及相应翻译方案

S→aAb　{print “1”}

S→a　{print “2”}

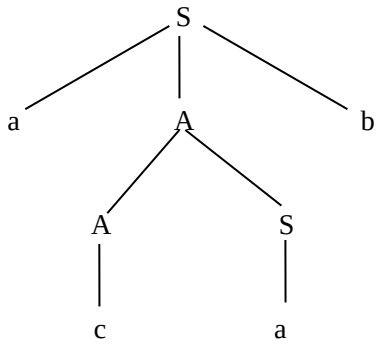
A→AS　{print “3”}

A→c　{print “4”}

输入 acab，输出是什么？

解答

对于这类题目，首先画出句子 acab 对应的语法树。每当用一个产生式归约时，则执行相应的语义动作。句子 acab 对应的语法树为：



第一个归约的产生式是 A→c，它相应的语义动作为 print “4”。所以，产生输出 4。
第二个归约的产生式是 S→a，它相应的语义动作为 print “2”， 产生输出 2。
第三个归约的产生式是 A→AS，它相应的语义动作为 print “3”，产生输出 3。
最后归约的产生式是 S→aAb，它相应的语义动作为 print “1”， 产生输出 1。
因此，输入 acab，输出是 4231。