第三版前言：

加入了 2012-2014 年真题，2011 年真题尚无。由于能力有限，未给出答案。感谢众多好心人提供真题！需说明的是，这些题目都是复试完后的童鞋回忆出来的，所以不免有些错误或不清楚的地方。所以，大家在做真题的时候，需要进行甄别，理性对待。编译可能第一次复习得时候比较难以理解，但是多复习几次后，就比较简单了，解题方法相对固定。

另外，2014 年南大计算机复试的笔试推荐教材有重大变化，大家最好是以当年的推荐教材为主，其他教材为辅进行复习。

此外，2014 年南大计算机复试的笔试是将两科的试题各分在两张试卷上，分开考的，先考编译（之前的几年中，两科的试题是在一张试卷上，一起考）。离散所占分值依然是 80 分，编译 70 分。考试总时间依然是 3 小时，编译的答题时间最多 1.5 小时。

jas923

2014.4.8

第二版前言：

加入 2010 年试题类型提示,并将有把握的 2009 年的试题答案予以补充。略微修改了答案明显有错误的题目。由于 2010 年编译原理题目太长，本人记忆力有限，所以只记住了题型与变化的地方。

编译原理比离散数学简单的多，而且难道不大。主要是记忆的成分比较多，所以大家一定要加强训练，多做点题目。编译原理试题和南大 CS 本科的期末试题有很大联系，故同学们复习的时候能够找到南大 CS 本科试卷，这样复习一定能够事半功倍！

特别感谢冷城学长提供原始文档。希望大家能够南大 CS 金榜题名！

Zyszys3

2010 年 8 月 19 日

前言

本文收集了 1997 年到 2007 年和 2009 年南京大学研究生入学考试科目《编译原理》的试卷、1997 年到 2007 年编译原理参考答案以及 04、05、06 年南京大学计算机系《编译原理》期末考试试卷。2008 年试题未找到，1997 年到 2007 年试题给出本人所做的答案，答案在考研复试准备中与同学进行核对过，可以确保大部分答案的正确性，2009 年试题答案未与同学核对，不能确保答案正确性，故不在此给出。

南京大学在 1997 至 2004 年《编译原理》作为初试的一部分，2005 年开始作为复试科目，满分一般为 70 分。

编译原理作为复试主要考理解能力和分析能力。重点在于文法的分析、翻译方案和优化方案。其中翻译方案刚接触时难度较大，深入理解后则很容易解决。

复习的推荐书目为吕映芝的《编译原理》，清华大学出版社。张幸儿的书用来参考，了解南大的编译原理中符号的书写规则。对于翻译方案一章则主要以张幸儿的书为主，吕映芝的讲解深奥难懂，而且不对应南大试卷的重点。翻译方案一章中回溯的方案一般不会考（只是一般不会考，如果要考到也没办法）。

冷城

2009 年 7 月

1997 年

六．填空

1. 语言 L 的形式定义是 L(G[2]) = _____

2. 当把=>+看作关系时，=>+是关系=>的_____

3. 扫描程序自动生成的实质是_____

4. 对原程序进行编译室，可以有以下几种中间表示：_____、_____、_____与_____等。

5. 算符优先分析技术分析过程中，每步直接归约的是当前句型中的_____

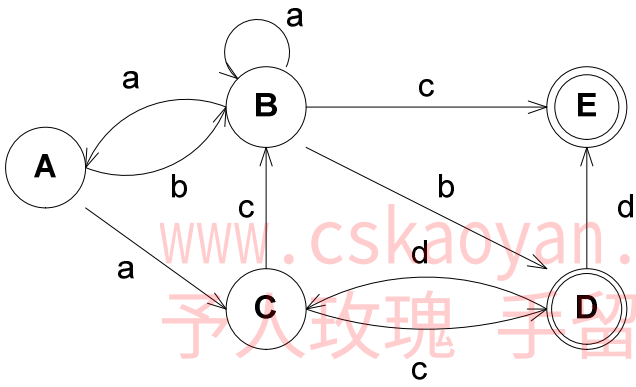
七. 简要回答下列问题

1. 试简述二义性文法的概念。

2. 试简要说明栈在编译实现中的作用，并列举至少 3 种不同的用途。

3. 试简要说明运行时刻存储管理的策略，对照某种程序设计语言，说明策略的特点。

八．试为下列状态转换图的相应的 NFA 确定化，要求写出关键步骤。



初试态： B、A

终止态： E、D

九．试用某种高级程序设计语言为下列文发 G[M]：

G[M]： M ::= Mab | Mac | dM | e　　B ::= bB | b

写出递归下降识别程序，要求：指名所用程序设计语言，变量说明等，说明性信息可以省略不写，但所写程序必须符合所用程序设计语言的语法规则。

十．试对下列程序片断进行源程序级的优化，指名何处进行何种优化。

```
s := 0;
for i := 1 to 100 do
begin
    a := ( x + y ) * i;
    b := ( x - y ) * i;
    s := s + a + 2 * b;
end;
```

十一．试应用 LR 分析技术识别输入符号串 ababab 是否下列文法 G[Z]的句子。按下列格式写出分析步骤。

步骤　栈　其余输入符号　动作　说明

G[Z]:

1. Z ::= CbBA　2. A ::= Aab　3.A ::= ab　4.B ::= c　5.B ::= Db　6.C ::= a　7.D ::= a

分析表见下页

	ACTION			GOTO				
	a	b	#	Z	A	B	C	D
0	S3			1			2	
1			acc					
2		S4						
3		r6						
4	S8					5	6	7
5	S11				10			
6	r4							
7		S9						
8	r6	r7						
9	r5							
10	S13		r1					
11		S12						
12	r3		r3					
13		S14						
14	r2		r2					

1997 年答案：

六．1. {x|s=>*x，其中 S 为文法识别符号，且 $x \in V_T$}

2. 传递性闭包

3. LR(k)分析表的自动生成

4. 逆波兰式，三元式，抽象语法树，四元式

5. 最左素短语

七．1. 如果对于某文法的同一个句子，存在两个不同的语法树，则称该句子试二义性的，包括二义性句子的文法成为二义性文法。

2. 提供一个后进先出数据结构实现类型。

用途 1：存放标示符，实现静态作用域法则和最接近套嵌约定。

用途 2：实现递归下降分析技术。

用途 3：实现简单优先分析技术与算符优先分析技术。

用途 4：实现 LR 分析技术。

用途 5：作为整个程序数据空间，用于可变数据及管理过程的控制信息。

3. 运行时存储管理策略有：静态存储分配、栈式分配和堆式分配。

静态存储分配：在编译时能确定目标程序运行中所需全部数据空间大小，编译时安排好运行时全部数据空间，确定每个数据对象的存储位置。

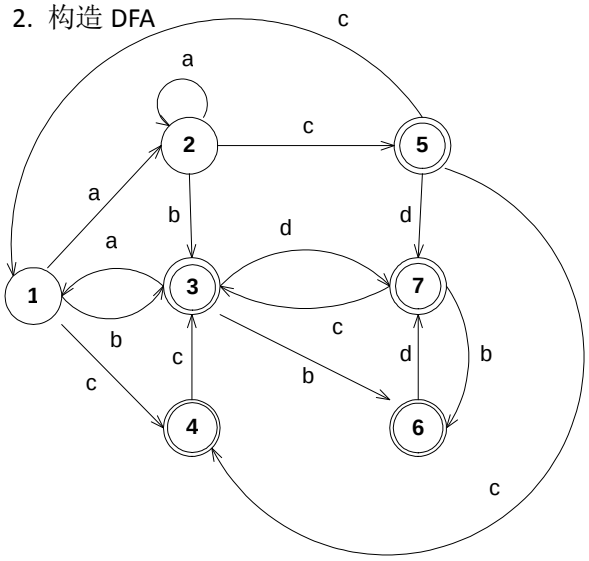
栈式分配：将整个程序数据空间设计为一个栈，每当调用一个过程时，它所需要的数据空间就分配在栈顶，每当过程工作结束时就释放这部分空间。

堆式分配：程序运行时有一个大的存储空间，每当需要时从这片空间中借用一块，不用时再退还。数据对象随即创建和消亡。每当过程调用结束时，局部变量值可保存，被调用生存周期可比调用生存周期更长。

八．1. 子集法构造 DFA

	a	b	c	d
AB[1]	ABC[2]	BD[3]	E[4]	/
ABC[2]	ABC[2]	BD[3]	BDE[5]	/
BD[3]	AB[1]	D[6]	E[4]	CE[7]
E[4]	/	/	/	/
BDE[5]	AB[1]	D[6]	E[4]	CE[7]
D[6]	/	/	/	CE[7]
CE[7]	/	/	BD[3]	/

2. 构造 DFA

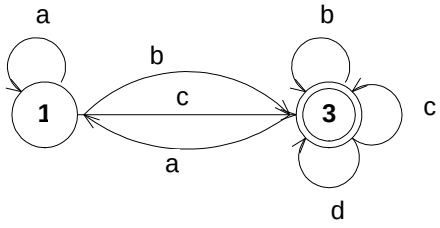


3. DFA 最小化

A1={1, 2}　　A2={3, 4, 5, 6, 7}

{1, 2}a⊂A1　　{3, 4, 5, 6, 7}a⊂A1

{1, 2}b⊂A2　　{3, 4, 5, 6, 7}a⊂A2



九．1. 消除左递归

M ::= Ma(b|c) | dM | e

改为：

M ::= (dM|e)M’

M’ ::= a(b|c)M’| ε

即：

M ::= dMM’ | eM’

M’ ::= aM’’ | ε

M’’ ::= bM’ | cM

2. 写出递归下降子程序

```
void GetSymbol(){…}
void Error(){…}
void M()
{
    if(sym == ‘d’)
    { GetSymbol();   M();   M1();}
    else if(sym == ‘e’)
    {GetSymbol();   M1();}
    else    Error();
}
void M1()
{
    if(sym == ‘a’)
    {GetSymbol();   M2();}
    else   return;
}
void M2()
{
    if(sym == ‘b’)
    {GetSymbol();   M1();}
    else if(sym == ‘c’)
    {GetSymbol();   M();}
    else Error();
}
void main()
{
GetSymbol();
M();
}
```

十．1. 消减强度

a := (x + y) * i 可改为： a := 0; a := a + x + y

b := (x – y) * i 可改为： b := 0; b := b + x – y

S := S + a + 2 * b 可改为： S := S + a + b + b

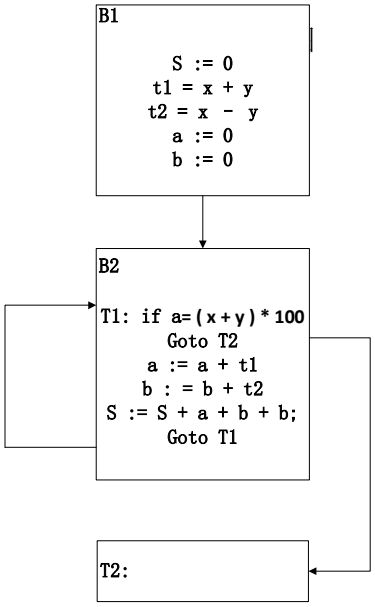
2. 消除归纳变量

i ≤ 100 可改为： a ≤ (x + y) * 100

3. 循环体内代码外提

重复计算了 x + y, x – y

所以设 t1 = x + y; t2 = x – y 并外提出循环体



十一.

步骤	栈	其余输入符号串	动作	说明
1	0	ababab#	S2	移入 a
2	0a3	babab#	r6	规约 a
3	0C2	babab#	S4	移入 b
4	0C2b4	abab#	S8	移入 a
5	0C2b4a8	bab#	r7	规约 a
6	0C2b4D7	bab#	S9	移入 b
7	0C2b4D7b9	ab#	r5	规约 Db
8	0C2b4B5	ab#	S11	移入 a
9	0C2b4B5a11	b#	S12	移入 b
10	0C2b4B5a11b12	#	r3	规约 ab
11	0C2b4B5A10	#	r1	规约 CbBA
12	0Z1	#	acc	接受

1998 年

六．简述概念过给出相应定义

1. 字母表 A 之正闭包的定义
2. Chomsky 文法的定义及其分类之原则
3. 活动记录及其作用
4. 编译程序语义分析的功能
5. 代码优化的概念及与循环相关的优化之种类

七．试为正则表达式 0{01}1 构造相应的确定有穷状态自动机。要求：写出各关键步骤。

八．试为文法 G[S]：

S ::= S;A | A　　A ::= i := E

E ::= E+F | F　　F ::= (E) | i

构造相依的 LL(1)分析表。

九．试为 PASCAL 语言 WHILE 语句：

WHILE E DO S

构造生成目标代码（无需回填）的翻译方案。

说明：　（1）无需给出关于表达式的翻译方案；

（2）必要时作简要的相应说明。

十．给出下列程序片段：

```
max := A[1];
FOR I := 2 TO 100 DO
    IF A[i] > max THEN max := A[i]
```

的响应三元式序列。

说明：　(1)必须考虑到每个机器字占 4 个字节；

(2)必要时对所使用的符号作简要说明。

1998 年答案

六．1. 【略】

2. Chomsky 文法 G 是一个四元组(V_N, V_T, P, Z)，V_N 为非终结符号集合；V_T 为终结符号集合；P 为非空重写规则；Z 为识别符号，且 $Z \in V_N$。

0 型文法：对于文法 G，P 中规则具有如下形式：U ::= V, $U \in V^*$, $V \in V^*$

1 型文法：对于文法 G，P 中规则具有如下形式：xUy ::= xUy, $U \in V_N$, $x,y \in V^*$, $U \in V^*$

2 型文法：对于文法 G，P 中规则具有如下形式：U ::= u, $U \in V_N$, $u \in V^*$

3 型文法：对于文法 G，P 中规则具有如下形式：U ::= T 或 U ::= WT, $T \in V_T$, $u,w \in V_N$

3. 【略】

4. 语义分析功能有：

①确定类型：确定标示符所关联数据对象的数据类型。

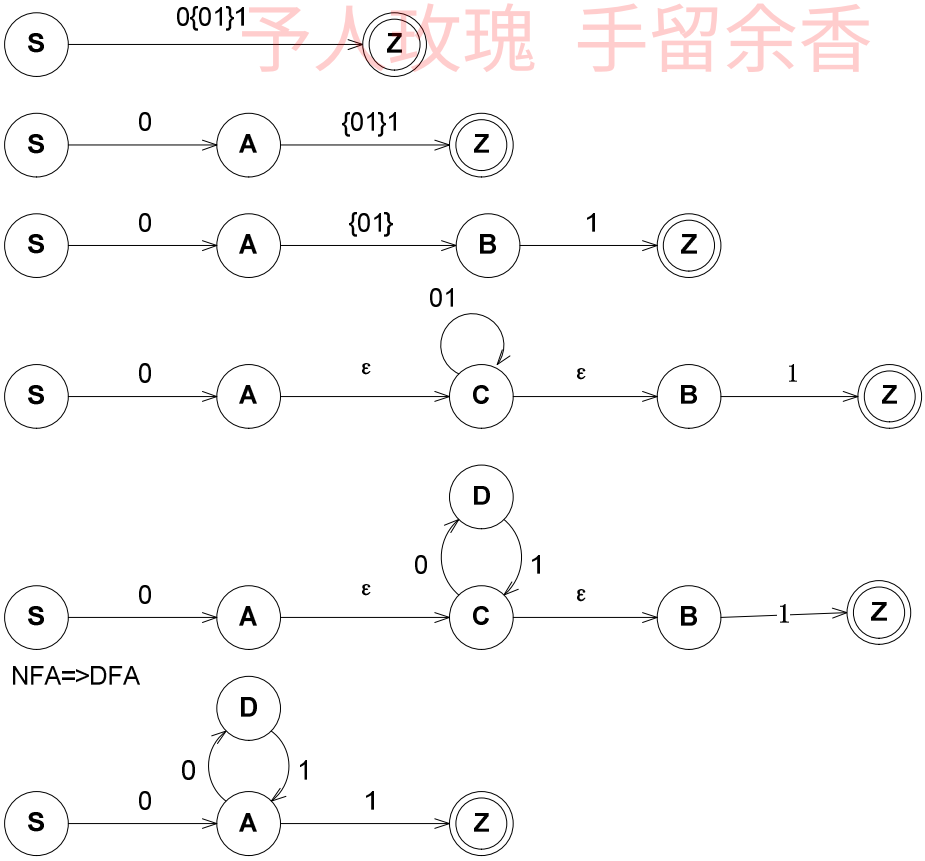
②类型检查：按照语言的类型规则，对运算及进行运算的运算分量进行类型检查，检查运算的合法性与运算分量类型的一致性。

③识别含义：根据程序语言的语义定义确认程序中各构造成分组合到一起的含义，并做相应的语义规则。

④其他一些静态语义检查，如控制流程检查。

5. 代码优化是编译时为改进目标程序质量而进行的各项工作，改进质量以提高两方面的效率，即时间效率和空间效率。与循环相关优化种类有：循环不变量表达式外提，归纳变量删除，计算强度消减等。

七．



八．1. 消除左递归

S ::= AS’

S’ ::= ;AS’ | ε

A ::= i := E

E ::= FE’

E’ ::= +FE’ | ε

F ::= (E) | i

2. 求 FIRST 集

FIRST(S) = { i }

FIRST(S’) = { ; ε }

FIRST(A) = { i }

FIRST(E) = { (i }

FIRST(E’) = { + ε }

FIRST(F) = { (i }

3. 求 FOLLOW 集

FOLLOW(S) = { # }

FOLLOW(S’) = { # }

FOLLOW(A) = { ; #}

FOLLOW(E) = {) ; # }

FOLLOW(E’) = {) ; # }

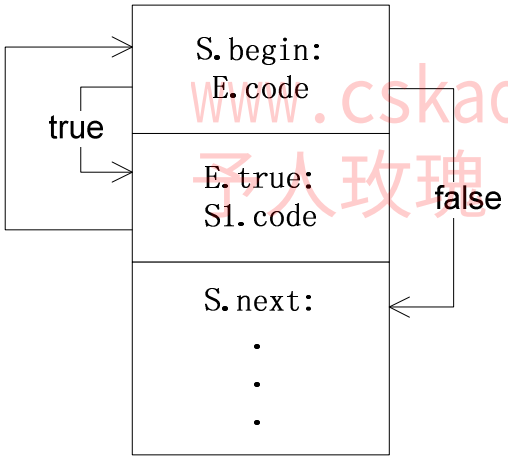
FOLLOW(F) = { +) ; # }

4. 构造相应的 LL(1)分析表

	;	i	:=	+	(	)	#
S		AS’					
S’	;AS’						ε
A		i := E					
E		FE’			FE’		
E’	ε			+FE’		ε	ε
F		i			(E)		

九．1. 重写规则：S→WHILE E DO S1

2. 画出目标代码示意图：



3. 写出翻译方案，中间代码用四元式表示：

```
{   S.begin = newlable;
    E.true = newlable;
    S.next = newlable;
    E.flase = S.next;
    S1.next = S.begin;
    S.code = gencode( s.begin':' )
             || E.code
             || gencode(CJ=, E.place, '1#', E.true )
             || gencode(GOTO, _, _, E.place)
             || gencode(E.true':')
             || S1.code
             || gencode(GOTO, _, _, S1.next)
             || gencode(S.next':')    }
```

注释：

newlable 用于产生一个新标号。

gencode(op, S1, S2, d)是生成目标指令函数，本题中使用四元式表示。

||是并置操作，将两个字符串并置

CJ=是相等判断操作，是则转向四元式最后一项，否则不执行

GOTO 是无条件跳转操作

十.

(1)	(	=[],	A,	1	)
(2)	(	:=,	(1),	max	)
(3)	(	:=,	2,	i	)
(4)	(	<=,	i,	100	)
(5)	(	GOF,	(14),	(4)	)
(6)	(	*,	i,	4	)
(7)	(	=[],	A,	(6)	)
(8)	(	>,	(7),	max	)
(9)	(	GOF,	(11),	(8)	)
(10)	(	:=,	(7),	max	)
(11)	(	+,	i,	1	)
(12)	(	:=,	i,	(11)	)
(13)	(	GO,	(4),	__	)

1999 年

六．简要回答下列问题

1. 试简述压缩了的文法的概念
2. 试简述运行状态转换图的概念
3. 试简述语法分析程序自动生成的基本思想
4. 试简要阐述语法制导翻译的概念
5. 试以 PASCAL 语言为例，简要说明运行时刻动态存储管理的必要性与策略

七．试归纳证明上下文无关文法的下列性质：

设 x =>* y，若 x 的首符号试终结符，则 y 的首符号也是终结符；反之，若 y 的首符号是非终结符，则 x 的首符号也是非终结符。

八．试为正则文法 G[C]：A :: = Ba | a　　B ::= Ab | b　　C ::= Ab | Ba　写出相应的正则表达式 e，使得|e| = L(G[c])。（方法不论，但解题应规范，需给出关键步骤）

九．设文法 G[A]：

G[A]：　A ::= iB * e　　B ::= SB　B ::= ε

S ::= [eC]　　S ::= .i　　C::= eC　C ::= ε

试用 LL(1)分析技术识别符号串 i.i*e 是否该文法的句子。以下列格式写出识别过程：

步骤　　栈　　输入符号串　　输出

该文法的 LL(1)分析表如下所示。

	i	*	e	[	]	#
A	A ::= iB * e					
B		B ::= ε		B ::= SB		B ::= SB
C			C::= eC		C ::= ε	
S				S ::= [eC]		S ::= .i

十．试为下列程序片段写出相应的四元式序列。

```
alldone := true
FOR index := 1 TO n – j DO
BEGIN
    i := index + 1;
    IF inrow[index] < inrow[i] THEN
    BEGIN
        temp := inrow[index];
        inrow[index] := inrow[i];
        inrow[i] := temp;
        alldone := false;
    END
END
```

注意：　(1)解答应按规范步骤写出；

(2)必要时对所使用的符号作简要说明。

1999 年答案

六．1. 如果文法 G 无多余规则，即每个规则 U ::= u 都满足：

条件 1、Z=>*xUy，其中 x,y∈V*，Z 为识别符号。条件 2、U=>+t，t∈V_T^+

2. 状态转换图是专门用来识别符号，包含有穷多个状态，除了开始状态不代表任何非终结符号外，每个节点都代表文法的非终结符号，弧上的标记指明在射出弧的节点状态下可能出现的输入字符或字符类的有向图。

3. LR(k)分析程序自动生成实质上是分析表的自动构造，首先编写分析表自动生成程序，用以将文法转化为分析表，然后用含次分析表的驱动程序识别输入符号串。

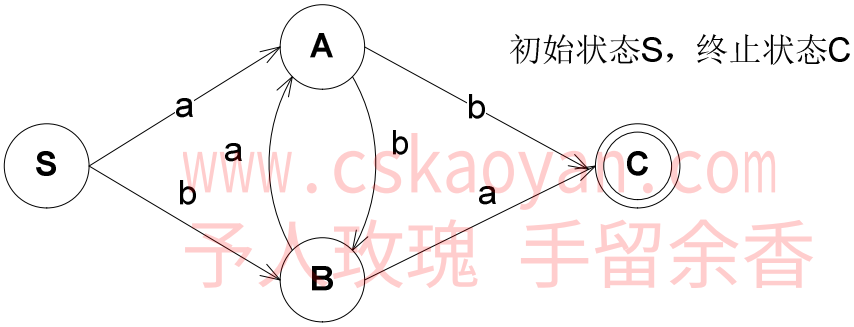
4. 语法分析中，随着分析的步步进展，根据每个产生式所定义的语义子程序或语义规则描述的语义动作，进行翻译的办法称作语法制导翻译。

5. 在程序需要使用递归过程、可变数组或允许用户自由申请和释放空间，则需要采用动态存储管理，一般有栈式和堆式两种。

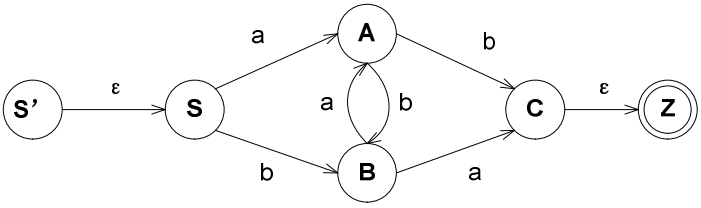
七．见书本【定理 2.8】

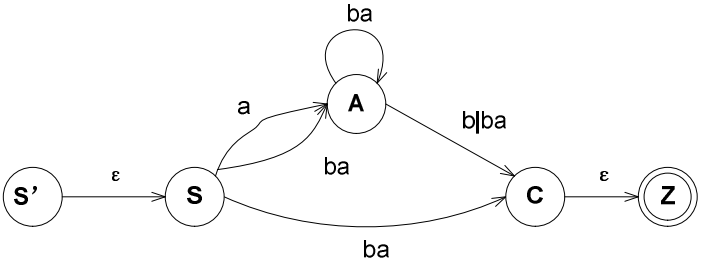
八．由题得 C 为终止状态

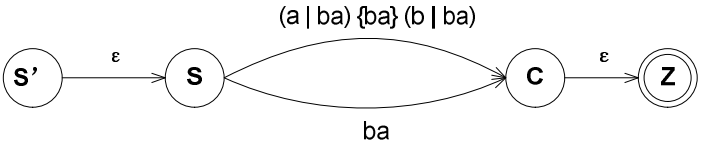
1. 画出 NFA

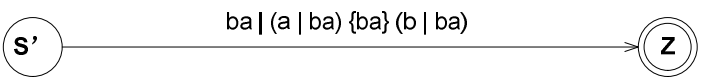


2. NFA 推导正则表达式









3. 正则表达式为 e = ba | (a | ba) { ba } (b | ba)

九.

步骤	栈	输入符号串	输出
0	#A	i.i*e#	A ::= iB*e
1	#e*Bi	i.i*e#	
2	#e*B	.i*e#	B ::= SB
3	#e*BS	.i*e#	S ::= .i
4	#e*Bi.	.i*e#	
5	#e*Bi	i*e#	
6	#e*B	*e#	B ::= ε
7	#e*	*e#	
8	#e	e#	
9	#	#	acc

十.

```
(1)  (   :=,      true,    __,     alldone  )
(2)  (   :=,      1,       __,     index    )
(3)  (   —,       n,       j,      T1       )
(4)  (   NJ<=,    index,   T1,     (21)     )// NJ<=为小于等于比较，假时跳转到第四项
(5)  (   +,       index,   1,      T2       )
(6)  (   :=,      T2,      __,     i        )
(7)  (   =[],     inrow,   index,  T3       )
(8)  (   =[],     inrow,   i,      T4       )
(9)  (   NJ<,     T3,      T4,     (18)     )//NJ<为小于比较操作，为假时跳转到第四项
(10) (   =[],     inrow,   index,  T3       )
(11) (   :=,      T5,      __,     temp     )
(12) (   =[],     inrow,   i,      T6       )
(13) (   []=,     inrow,   index,  T7       )
(14) (   :=,      T6,      __,     T7       )
(15) (   []=,     inrow,   i,      T8       )
(16) (   :=,      temp,    __,     T8       )
(17) (   :=,      false,   __,     alldone  )
(18) (   +,       index,   1,      T9       )   //T9 = T2
(19) (   :=,      T9,      __,     index    )
(20) (   GO,      __,      __,     (4)      )   //GO 为无条件跳转指令
```

2000 年

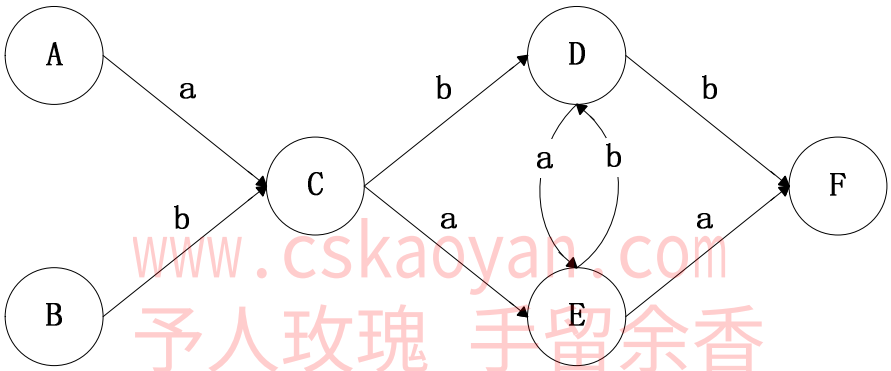
六．简要回答下列问题。

1. 试用至少两种不同的形式表示法描述由 7/9 的一切精度的近似值所组成的集合。
2. 试简述二义性概念。
3. 试简述有穷状态自动机与正则表达式的等价性概念。
4. 移入—规约法是否是一种语法分析技术？请简述理由。
5. 设有 PASCAL 型函数说明如下：

```
FUNCTION F( n: integer; m : integer; A : ARRAY[-1…n, 10…m] OF integer ) : real;
BEGIN
    …
END
```

试写出与 F 相关联的类型表达式。

七．试为下图所示的状态转换图写出相应的有穷状态自动机。



八．试用 PASCAL 或 C 语言为下列文法 G[S]：

G[S]:　S ::= TP　　P ::= aS | ε　　T ::= QR

R ::= T | ε　　Q ::= aSb | c

写出递归下降识别程序。

说明：　(1)指明所用程序设计语言；

(2)类型说明及底层实现细节。

九．试为 PASCAL 语言 REPEAT 语句：

```
REPEAT S UNITL E
```

设计生成虚拟目标代码（无需回填）的语法制导定义。

说明：　(1)假定关于表达式的目标代码仅计算表达式的值；

(2)无需给出关于表达式的语法制导定义；

(3)必要时对所引进符号给出简要的。

十．设有 PASCAL 型程序片段如下：

```
VAR A,B : ARRAY[1…5, 1…5] of integer;  i,j : integer;
FOR i := 1 TO 5 DO
    FOR j := 1 TO 5 DO
    GEGIN
        A[i,j] := i + j;
        B[i,j] := 11 – (i + j);
    END
```

试写出相应的四元式序列，指出何处可进行何种优化。

说明：　(1)解题步骤必需规范，写出各关键步骤；

(2)假定每个整型量占 1 个字节，且数组第一个元素的存储地址用数组名表示；

(3)无需重写最终结果，可仅在相应的四元式处指明变化；

(4)必要时对所引进符号给出简要的解释。

2000 年答案

六.

1. (1) 0.{7}8 | 1　　(2) S -> AB | 1　　A -> 0.　　B -> B8 | 7

2. 如果对于某文法的同一个句子存在两个不同的语法树，则称该句子是二义性的，包含有二义性句子的文法称为二义性文法。

3. 一个正则表达式 e 能构造一个有穷状态自动机 A，使得 L(A) = |e|，一个有穷状态自动机 A 同样能构造一个表达式，使得|e| = L(A)，那么正则表达式与有穷状态自动机可以定义或接受相同正则集，这称为正则表达式与有穷状态自动机的等价性。

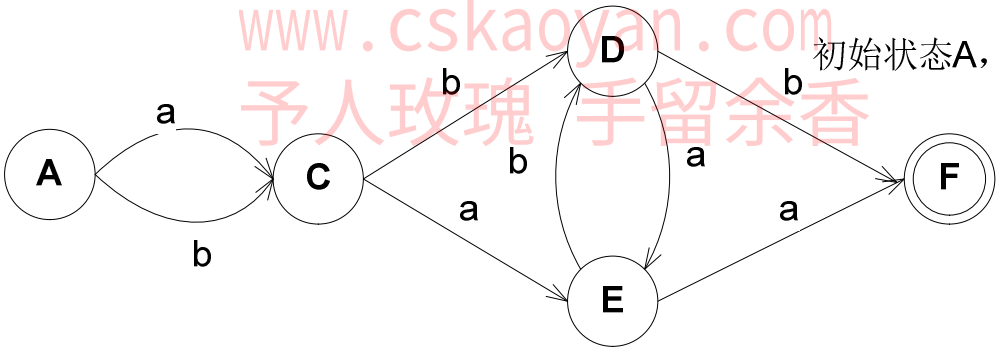
4. 是，对于句柄的移入—规约适应某些文法。（不确定）

5.【略】

七. 1. 构造状态转换表

	a	b
AB	C	C
C	E	D
E	F	D
D	E	F
F	/	/

2. 构造 DFA



八.

```
GetSymbol(){…};
Error(){…};
void S()
{
      T();  p();
}
void P()
{
      if(sym == 'a'){  GetSymbol();  S();  }
      else       return;
}
void T()
{
```

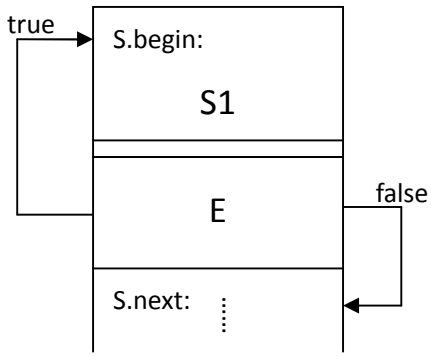
FIRST(S) = { a, c }
FIRST(P) = { a }
FIRST(T) = { a, c }
FIRST(R) = { a, c, ε }
FIRST(Q) = { a, c }

```
Q();        R();
}
void R()
{
    if(sym == 'a' || sym == 'c')     T();
    else        return;
}
void Q()
{
    if(sym == 'a')
    {
        GetSymbol();  S();
        if(sym == 'b'){ GetSymbol();  return;   }
        else        Error();
    }
    else if(sym == 'c'){  GetSymbol();  return;   }
    else Error();
}
void main()
{
    GetSymbol();        S();
}
```


九. 1. 重写规则

S -> REPEAT S1 UNTIL E { S.begin = newlable; S.next = newlable;
E.true = S.begin; E.false = S.next;
S.code = genquade(S.begin':')
|| S.code
|| E.code
|| genquade(JC=, E, '1#', E.true)
|| genquade(GOTO, _, _, E.false)
|| genquade(S.next':') }

2. 画出目标代码示意图



3. 写出语义规则

4. 注释

newlable 用于产生一个新标识符；

genquade()用于产生一个四元式；

JC=是相等比较操作，相等则转向四元式第四项

十. 1. 未优化的四元式：

(1)	(	:=,	1,	_,	i)
(2)	(	>,	i,	5,	(18))
(3)	(	:=,	1,	_,	j)
(4)	(	>,	j,	5,	(16))
(5)	(	+,	i,	j,	t1)
(6)	(	[]=,	A,	i,	t2)
(7)	(	[]=,	t2,	j,	t3)

(8) (&:=, t1, _, t3)
(9) (+, i, j, t4)
(10) (—, 11, t4, t5)
(11) ([]=, B, i, t6)
(12) ([]=, t6, j, t7)
(13) (&:=, t5, _, t7)
(14) (+, j, 1, j)
(15) (GO, _, _, (4))
(16) (+, i, 1, i)
(17) (GO, _, _, (2))
(18) ……

2. 合并后的四元式：

(5)与(9)是公共子表达式，可以合并。

(1) (:=, 1, _, i)
(2) (>, i, 5, (17))
(3) (:=, 1, _, j)
(4) (>, j, 5, (15))
(5) (+, i, j, t1)
(6) ([]=, A, i, t2)
(7) ([]=, t2, j, t3)
(8) (&:=, t1, _, t2)
(9) (—, 11, t1, t5)
(10) ([]=, B, i, t6)
(11) ([]=, t6, j, t7)
(12) (&:=, t5, _, t7)
(13) (+, j, 1, j)
(14) (GO, _, _, (4))
(15) (+, i, 1, i)
(16) (GO, _, _, (2))
(17) ……

2001 年

四．简要回答下列问题

1. 简要说明语法分析树有何应用？给出至少 3 种应用。

2. 简要说明属性文法的概念，并说明它在编译实现中起什么作用？

五．设有下列 LL(1)分析表（其中 A 是根符号），写出输入串# var id , id : int #的 LL(1)分析过程，例如该分析过程的开始时刻和第 9 时刻如下：

时刻	分析栈	当前输入符	输出产生式
开始	#A	var	A → var B
.	.	.	.
.	.	.	.
9	#B'F:D	id	D → ED
.	.	.	.
.	.	.	.

分析表	var	id	;	,	int	:	array	const	[	]	of	#
A	A→var B											
B		B→CB'										
B'			B' →;B									B'→ ε
C		C→D:F										
D		D→ED'										
D'				D' →,D		D' → ε						
E		E→id										
F					F→int		F→array[G]ofF					
G								G→const				

六．下列程序段的功能是把 n 个元素的数组之“对称元素对”进行交换。试为其写出相应的四元式序列，并指明何处可能进行何种优化。

PASCAL 语言

```
FOR i := 1 TO n DIV 2 DO
BEGIN
    t := A[i];
    A[i] := A[n-i+1];
    A[n-i+1] := t
END
```

或

C 语言

```
FOR( i = 1; i <= n/2; i ++)
{
    t = A[i];
    A[i] := A[n-i+1];
    A[n-i+1] := t
}
```

说明：

1. PASCAL 与 C 语言可任选一种，但解题必须规范。

2. 无需给出优化的结果，要求的仅仅是“指明”。

3. 必要时对所用的符号加以说明。

2001 年答案

四．1. 一、判断文法是否具有二义性；二、寻找字符串符号之间优先关系；三、寻找句柄或最左素短语；四、构造节点属性依赖图。

2. 对于某个压缩了的上下文无关文法，当把每个文法符号联系于一组属性，且让该文法中的重写规则附加以语义规则时，称该上下文无关文法为属性文法。属性文法是上下文无关文法的扩充，是把上下文无关语言同上下文有关语言的语义信息相结合的形式定义，是语义分析形式定义的有效手段。

五．

时刻	分析栈	当前输入符	输出产生式
开始	#A	var	A -> var B
1	#B var	var	
2	#B	id	B -> CB’
3	#B’C	id	C -> D:F
4	#B’F:D	id	D -> ED’
5	#B’F:D’E	id	E -> id
6	#B’F:D’id	id	
7	#B’F:D’	,	D’ -> ,D
8	#B’F:D,	,	
9	#B’F:D	id	D -> ED’
10	#B’F:D’E	id	E -> id
11	#B’F:D’id	id	
12	#B’F:D’	:	D’ -> ε
13	#B’F:	:	
14	#B’F	int	F -> int
15	#B’int	int	
16	#B’	#	B’ -> ε
17	#	#	acc

六．C 语言版

```
(1)  (   =,   1,   _,   i    )
(2)  (   /,   n,   2,   T1   )   ①可以合并常量
(3)  (   <=,  i,   T1,  (5)  )
(4)  (   GO,  _,   _,   (17) )
(5)  (   =[],  A,   i,   T2   )
(6)  (   =,   T2,  _,   t    )
(7)  (   —,   n,   i,   T3   )   ②n+1 常量可合并
(8)  (   +,   T2,  1,   T4   )
(9)  (   =[],  A,   T4,  T5   )
(10) (   []=,  A,   i,   T6   )
(11) (   =,   T5,  _,   T6   )
(12) (   —,   n,   i,   T7   )  } ③与(7)(8)是公共子表达式，可消除
(13) (   +,   T7,  1,   T8   )  }
```

(14) ([]=, A, T8, T9)
(15) (=, t, _, T9)
(16) (+, i, 1, i)

2002 年

八．简要回答下列各题：（3 题共 10 分）

1. 给出 Chomsky 文法的形式定义，并简述其分类同程序设计语言之设计与实现的关系。(4 分)

2. 简述词法分析程序的基本功能试什么？现在让你实现词法分析程序（非自动生成且作为独立的一趟），你将如何考虑？请简述你的方案（实现要点）。(4 分)

3.试简述语义分析部分的基本功能。（2 分）

九．（10 分）

1. 写出表达式 A*(B*C-A)*B+C*D 的逆波兰表达式，三元式和四元式。（7 分）

2. 给定下列四元式，指出其中可消除的公共子表达式。（3 分）

+	a	b	c
—	c	d	b
+	a	d	b
—	c	d	c

十．简要说明编译程序对循环结构可能进行的优化，下列程序段：

PASCAL 语言	或	C 语言
y = A[n];		y = A[n];
FOR i := n—1 down to 0 DO		for(i = n—1; i >= 0 i— —)
y := y * x + A[i];		y := y * x + A[i];

可进行何种优化？写出相应的三元式序列。（10 分）

2002 年答案

八．1. Chomsky 文法是一个四元组(V_N, V_T, P, Z)其中 V_N 为非终结符号集合，V_T 为终结符号集合，P 为有穷非空的重写规则集合，Z 为开始符。

0 型文法对应短语结构语言；

1 型文法对应上下文相关语言；

2 型文法对应上下文无关语言；

3 型文法对应正则语言。

2. 基本功能是识别源程序中具有独立含义的最小语法单位——符号或单词。

将符号串自左至右依次输入，与文法规则相比较，若无违背文法规则，则接受。

3. 语义分析基本功能是在所写源程序上进一步分析其含义，在理解含义基础上为响应目标代买生成做准备（生成直接代码）或直接生成目标代码。

九．1. 逆波兰式：A B C * A 一 * B C D * +

三元式：(1) (*, B, C)
(2) (一, (1), A)
(3) (*, A, (2))
(4) (*, (3), B)
(5) (*, C, D)
(6) (+, (4), (5))

四元式：(1) (*, B, C, T1)
(2) (一, T1, A, T2)
(3) (*, A, T2, T3)
(4) (*, T3, B, T4)
(5) (*, C, D, T5)
(6) (+, T4, T5, T6)

2. (一, c, d, d) 与 (一, c, d, c)可消除公共子表达式：

改为： +, a, b, c
一, c, d, c
+, a, c, b

十．三元式为：

(1) (=[], A, n)
(2) (=, (1), y)
(3) (一, n, 1) //可用 n 代替 i，删除归纳变量
(4) (=, (3), i)
(5) (>=, i, 0)
(6) (JN, (5), (14))
(7) (*, y, x) //y = y * x + A[1]可进行强度消减
(8) (=[], A, i)
(9) (+, (7), (8))
(10) (=, (9), y)
(11) (一, i, 1)
(12) (=, (11), i,)
(13) (GO, _, (5))

2003 年

一．简述为什么自顶向下的语法分析技术不能处理具有左递归的文法。

二． 对于同一个文法，LALR(1)和 SLR(1)的分析表的状态个数相同，为什么前者的分析能力要比后者强？

三．给出下列赋值语句序列对应的四元式序列并且使用 DAG 图对得到的四元式序列进行优化

$$a = c - b * a - c + (a - c)$$

$$b = c - b * (a - c)$$

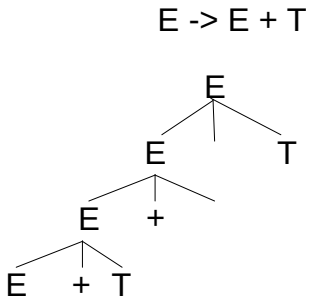
四．设 C 语言的 do-while 语句定义如下：

S::=do S while (E);

试为其设计目标代码结构，并写出相应的翻译方案。假定已为表达式 E 设计好翻译方案，执行 E 的目标代码将得到 E 的值。注意：必要时间略说明所引进符号的含义.

2003 年答案

一．若对左递归文法应用带回溯自顶向下分析技术情况将如右图所示；以 E 为目标，从 E+T 开始探测时，又将以 E 为目标，从 E+T 开始探测，如此循环往复永无止境。所以使用自顶向下分析技术时应避免左递归情况的出现。



二．SLR(1)方法解决冲突时，在对于规约项目 A→α. A→α.a 且 A∈FOLLOW(A)时会产生移入—规约冲突，SLR(1)默认将 A→α.进行规约，但规约后再移近 a 后，栈符号串串尾为 Aa，而 Aa 未必为规范举行活前缀的串尾。

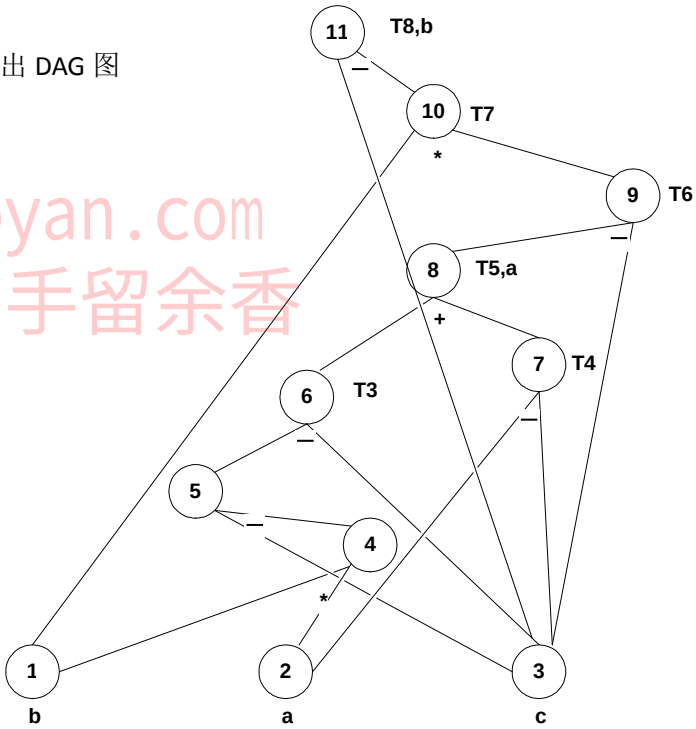
而 LALR(1) 在出现上述情况时，对于 A，若有 S→Aβ，则把 FIRST(β)作为 A→α 规约的搜索符，用以代替 SLR(1)分析中的 FOLLOW 集，因此若向前搜索符 FIRST∈(β)时，则进行规约，否则移入向前搜索符，这样有效解决了移入—规约冲突。

LALR(1)对同心集进行了合并，对项目集相同而向前搜索符不同的集进行合并，所以其项目集数与 SLR(1)的相同。

三．1. 写出未优化前的四元式：

(1) (*, b, a, T1)
(2) (—, c, T1, T2)
(3) (—, T2, c, T3)
(4) (—, a, c, T4)
(5) (+, T3, T4, T5)
(6) (=, T5, _, a)
(7) (—, a, c, T6)
(8) (*, b, T6, T7)
(9) (—, c, T7, T8)
(10) (=, b, _, T8)

2. 画出 DAG 图



3. 优化四元式。（好像没什么可优化的）

四．1. 重写规则：S→do S1
while E

2. 画出目标代码示意图：

3. 写出语义规则：

（2.3.答案同 2000 年第九题）

2004 年

六．回答下列问题【20 分】

1. 一个变异程序的前端部分÷语义分析之外，还包括哪两个部分？【4 分】
2. 在 LR 洗了分析技术中，LR(1)的使用范围一定大于 SLR(1)吗？【2 分】
3. 自底向上分析技术需要解决哪两个基本问题？【4 分】
4. 简单优先分析技术和算符优先分析技术哪个的适用范围广一些？为什么？【4 分】、
5. 为什么编译程序会对代码进行优化，而解释执行的语言的解释器一般不对代码进行优化？【2 分】
6. 存储分配策略中有静态存储分配、栈式存储分配和堆式存储分配。请问，全局变量使用哪种存储分配方式进行分配？C++语言中，由 new 创建的对象的内存使用哪种方式进行分配？【4 分】

七．证明题【12 分，每题 4 分】

给定文法 G[Z]:　　E ::= F | EF

F ::= aEb| ab

请证明：

1. 所有的 G 的句子中 a 的个数和 b 的个数一样多。
2. 如果 x 是 G 的句子，并且 y 是 x 的头，那么 y 中的 a 的个数不少于 b 的个数。
3. 如果符号串 x 满足下列条件：

a) x 只包含 a 和 b，

b) 且 a 的个数和 b 的个数一样多，

c) 且对于 x 的任何头 y，y 中 a 的个数不少于 b 的个数。

请证明 x 必然是 G 的句子

八．对于文法 G[E]:　　E ::= E + T | T

T ::= T*F | F

F ::= (E) | i

进行修改，使得可以用自顶向下技术进行分析。【8 分】

2004 年答案

六．1. 词法分析和语法分析。

2. 是。

3. 一、找出句柄；二、确定句柄规约到哪个终结符号。

4. 算符优先技术更广一些，简单优先分析技术考虑 $V = V_N \cup V_T$ 之间关系，而算符优先分析文法只考虑 V_T 之间的优先关系，所以有时间用简单优先技术分析有冲突，用算符优先文法分析则未必。算符优先分析技术的范围更广。

5. 因为编译出来的程序，每次运行都要进行编译，如果不做优化，则每次机器运行的太慢。而解释器只运行一次，以后使用不用优化。

6. 全局变量使用静态存储分配，new 创建对象内存使用堆式存储分配。

七．证明：

1. 写出关于 a，b 个数的属性文法

E ::= F　　E.anum = F.anum;　　E.bnum = F.bnum;

E ::= E1F　　E.anum = E1.anum + F.anum;　　E.bnum = E1.bnum + F.bnum;

F ::= aEb　　F.anum = E.anum + 1;　　F.bnum = E.bnum + 1;

F ::= ab　　F.anum = 1;　　F.bnum = 1;

由属性文法可以看出开始时 a 和 b 的个数相同，在操作中对 a 和 b 的操作相同，所以 a 和 b 的个数相同

2. ∵E 中 a 与 b 的个数相同，而由 F ::= aEb 与 ab 得到 a 必然在某个 b 之前，且他们是一一映射关系，∴有多少个 b 必然至少有多少个 a，∴y 中 a 的个数不少于 b 的个数。

3. 使用数学归纳法。

设 a 与 b 的个数为 k，k = 1 时句子为 ab，符合文法 G。

设 k≤n 时，满足 a)、b)、c)的 x 是 G 的句子，则此时 x 可化为 $F_{a1}F_{a2}……F_{ai}$；

则当 k = n + 1 时，x 由其中 a 与 b 相等的一些符号串拼接而成，且其中在任一尾字符串 a 与 b 不相等，可能的情况有　① $x = ax_ib$　拼接数为 1；

② $x = x1x2……xi$　拼接数为 i。

对于情况①，x_i 中 k≤n，则可以化为 x = aEb → x = F → x = E，是 G 的句子。

对于情况②，$x_r(1≤r≤i)$中 k≤n，则 x 可化为 $F_{1a1}F_{1a2}…F_{1ai}F_{2a1}F_{2a2}…F_{2ai}…F_{ia1}F_{ia2}…F_{iai}$ 为 G 的句型。

∴ k = n + 1 时也是 G 的句子，得证。

八．消除左递归：

E ::= TE’

E’ ::= +TE’ | ε

T ::= FT’

T’ ::= *FT’ | ε

F ::= (E) | i

2005 年

一．【20 分】请对下面的文法进行压缩变换（即消除无用规则）。并且消除所有文法的规则左递归

G[E]：　E ::= E + T | E – T | T

T ::= TMF | F | i

F ::= Fi

M ::= *

D ::= /

其中终结符号集合为{+,–,*,/} 非终结符号集合为{E,T,F,M,D}

二．【15 分】给定正则文法 G[A]：

B ::= Ba | a | Aa

A ::= Ba | Bb

构造出相应的 DFA，并将其最小化

三．【15 分】已知文法 G[E]：

E ::= E + T | T

T ::= TF | F

F ::= F* | (E) | a | b

给出 LR(1)项集{[F→(.E),+]}的闭包，以及该项集闭包相对于文法符号 E，T，F 的后继 LR(1)项集闭包

四．【20 分】已知带括号的四则运算表达式的语法如下：

G[E]:　E ::= E + T | T

T ::= T * F | F

F ::= (E) | i

请给出能够生成一个给定表达式的逆波兰表示的翻译方案

2005 年答案

一．1. 消除无用规则后得：

E ::= E+T | E—T | T

T ::= TMF | F | i

F ::= ε | Fi

M ::= *

2. 消除左递归后得：

E→ TE’

E’ → +TE’ | —TE’ | ε

T → (F | i) T’

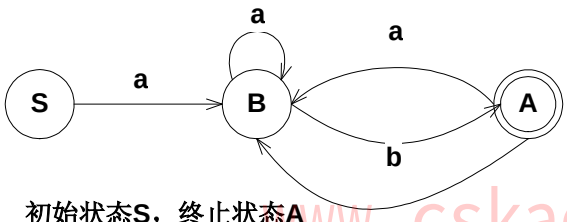
T’ → MFT’ | ε

F → F’

F’ → iF’ | ε

M → *

二．1. 画出 NFA

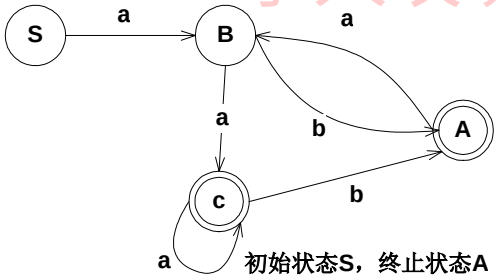


初始状态S，终止状态A

2. 构造状态转换表

	a	b
S	B	/
B	AB	A
AB	AB	A
A	B	/

3. NFA => DFA



初始状态S，终止状态A

4. 最小化

A1 = { S, B }　B1 = { A, C }

{ S }$_a$ ⊂ A1

{ B }$_a$ ⊂ B1

S 与 B 不是等价状态，可划分；

{ A }$_a$ ⊂ A1

{ C }$_a$ ⊂ B1

A 与 C 不是等价状态，可划分。

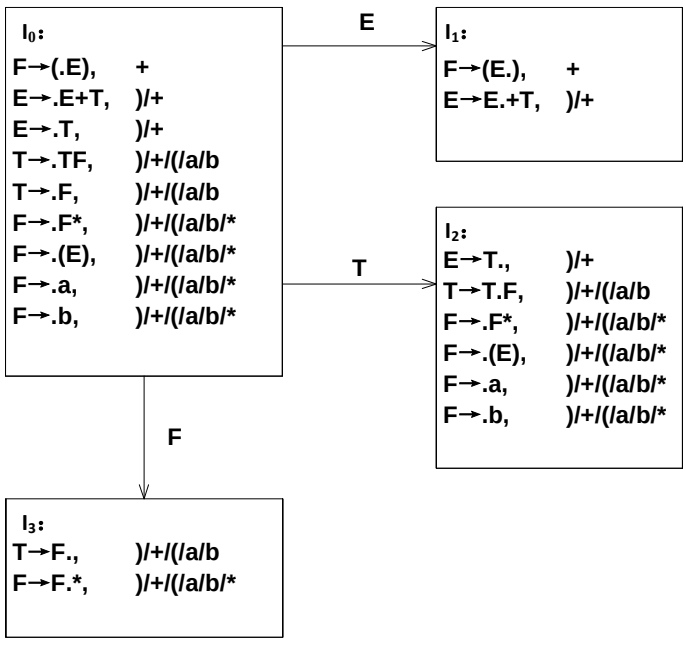
所以最终状态如左图不变。

三．

FIRST(E) = { (, a, b }

FIRST(T) = { (, a, b }

FIRST(F) = { (, a, b }



四．重写规则	翻译方案
E ::= E1 + T	{ E.code = E1.code || T.code || print (‘+’) }
E ::= T	{ E.code = T.code }
T ::= T1 * F	{ T.code = T1.code || F.code || print (‘*’) }
T ::= F	{ T.code = F.code }
F ::= (E)	{ F.code = E.code }
F ::= i	{ F.code = print (i.name) }

2006 年

一．已知文法 G[S]如下：

S -> if b then S | if b then S else S | v := e

其中 V_T = { if, b, then, else }　V_N = S，

a、 说明这个文法是二义性文法。【3 分】

b、 给出这个文法的 LALR(1)分析表，指出其中的冲突项。【10 分】

c、 指出如何修改这个分析表中的冲突项，使得扫描程序按照下面的规则扫描符号：”else”和最近的尚未匹配的”if”匹配。【7 分】

二．已知一个非常简单的模拟机具有 1K 个存储单元，每个单元存放一个整数，这个模拟机的指令格式如下：

OP　oprand1　oprand2　dest　或者　Mov　operand 1　dest

其中 OP 可以是+, −,×,/。而 operand1 和 operand2 是操作数，有两种形式：“#整数”和“整数”。第一种形式表示常量，而第二种形式表示存放在相应存储单元中的量。Dest 是一个整数，表示存储单元的编号。Mov 表示将 operand1 移动到 dest 指明的单元中。

例如：

- + #12 123 14 表示将常量 12 和第 123 存储单元中的值相加后，存放到 14 个单元中；
- Mov #1 0 表示将常量移动到第 0 个存储单元中；
- Mov 1 0 表示将第 1 个存储单元中的值移动到第 0 个存储单元。

现在给出一个简单语言的语法如下：

Program -> DecBody

DecBody -> ‘var’ VList ‘;’

VList -> Identifier ‘,’ VList | Identifier

Body -> Statement ‘;’ Body | Statement ‘;’

Statement -> Identifier ‘:=’ Expression

Expression -> Term | Expression ‘+’ Term | Expression ‘−‘ Term

Term -> Factor | Term ‘*’ Factor | Term ‘/’ Factor

Factor -> Identifier | Integer | ‘(‘ Expression ‘)’

这个语言的程序包括两个部分：声明变量(Dec)部分和运算(Body)部分。在变量声明中，程序员可以声明一系列的变量，而运算部分由一系列赋值语句组成。每个语句的左部是一个变量，而右部是一个带括号的四则运算表达式。语句中对变量的引用应该遵守先定义后使用的规则。

1、 请给出处理变量声明部分的翻译方案。要求：进行内存分配，检查是否有变量被重复定义。【10 分】

2、 请给出生成运算部分对应代码的翻译方案。【10 分】

3、 请给出一个翻译方案，判断每个赋值语句的右部是否为一个常量表达式。【10 分】

注意：Identifier 以及 Integer 在词法中定义。我们可以用 Identifier.text 和 Integer.text 来获取相应的字面值。

三．已知正则表达式 b{ab}a，请给出相应的最小化 DFA 和相应的正则文法。【10 分】

四．请给出如下的程序段（以 C 语言的语法给出）对应的四元式，并对其进行优化（假设 X, Y, Z, M 等变量以后都会被用到）。【10 分】

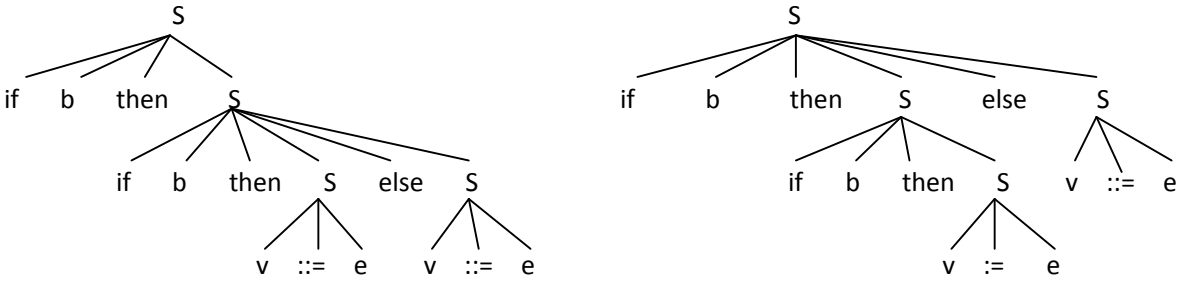
X = X * Z + Y / 2

M = X * Z + Y * Z

Z = Z * Y + Y / 2

2006 年答案

一．a、对于字符串 if b then if b then v ::= e else v ::= e 可构造如下两个不同语法树：



∴该文法是二义性文法。

b、增广型文法为：

S’ ::= S

S ::= if b then S

S ::= if b then S else S

S ::= v := e

构造 LALR(1)项目集簇：

I_0: S’→ .S, #

S→ .if b then S, #

S→ .if b then S else S, #

S→ .v := e, #

I_1: S’→ S., #

I_2: S→ if .b then S, #/else

S→ if .b hten S else S, #/else

I_3: S→ v .:= e, #/else

I_4: S→ if b .then S, #/else

S→ if b .then S else S, #/else

I_5: S→ v := .e, #/else

I_6: S→ if b then .S, #/else

S→ if b then .S else S, #/else

S→ .if b then S, #/else

S→ .if b then S else S, #/else

S→ .v := e, #/else

I_7: S→ v := e., #/else

I_8: S→ if b then S., #/else

S→ if b then S .else S, #/else

I_9: S→ if b then S else .S, #/else

S→ .if b then S, #/else

S→ .if b then S else S, #/else

S→ .v := e, #/else

I_{10}: S→ if b then S else S., #/else

S → I_1

if → I_2

v → I_3

→ acc

b → I_4

:= → I_5

then → I_6

e → I_7

S → I_8

if → I_2

v → I_3

S ::= v := e

S ::= if b then S

else → I_9

S → I_{10}

if → I_2

v → I_3

S ::= if b then S else S

画出 LALR(1)分析表：

状态	ACTION								GOTO
	if	b	then	else	v	:=	e	#	S
0	S_2				S_3				1
1								acc	
2		S_4							
3						S_5			
4			S_6						
5							S_7		
6	S_2				S_3				8
7				r_3				r_3	
8				r_1/S_9				r_1	
9	S_2				S_3				10
10				r_2				r_2	

0: S' ::= S
1: S ::= if b then S
2: S ::= if b then S else S
3: S ::= v := e

其中 ACTION[8][else]是冲突项，同时有归约和移入两种动作

c、把 ACTION[8][else]中冲突项 r_1/S_9 改为 S_9

二．1、变量声明部分的规则：　　翻译方案：

Dec→ 'var' VList ';'　　{ offset := 0 }

VList→ Identifier ',' VList1　　{ p = lookup(Identifier.text);
if (p != NULL)　ERROR(print("Identifer Redifinition!));
else { enter(Identifier.name, offset);
offset = offset + 1; }
}

VList→ Identifier　　{ p = lookup(identifier.text);
if(p != NULL)　ERROR(print("Identifier Redifinition!"));
else{ enter(Identifier.name, offset);
offset = offset + 1; }
}

2、运算部分代码的翻译：

Body →Statement ';' Body1　　{ Body.code = Statement.code
|| Body1.code }

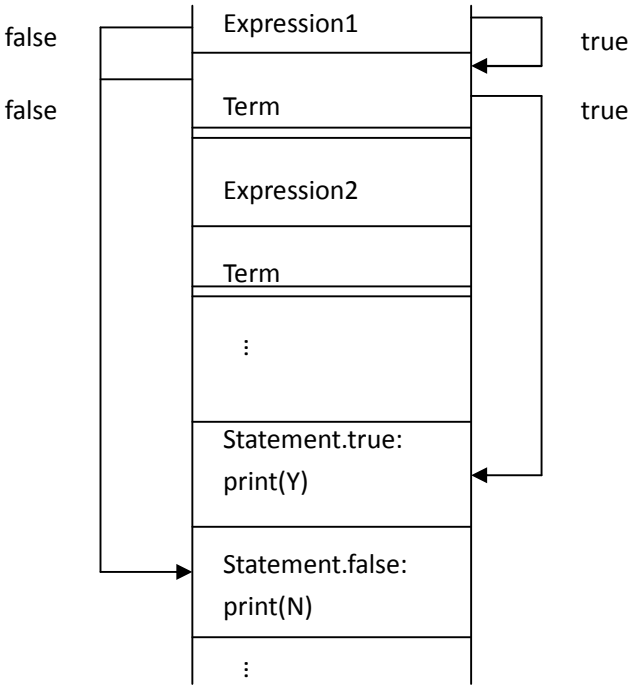
Statement →Identifier ':=' Expression　{ p = lookyp(Identifier.text);
if(p != NULL){
Statement.code = Expression.code
|| gencode(MOV, Expression.name, *p.place)
}
else ERROR();
}

Expression →Expression1 '+' Term　　{ Expression.place = newtemp;
Expression.code = Expression1.code
|| Term.code
|| gencode(+, Expression1.place, Term.place, Expression.place)
}

Expression →Expression1 '—' Term　　{ Expression.place = newtemp;
Expression.code = Expression1.code
|| Term.code

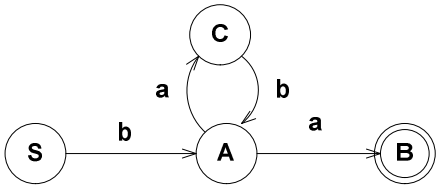
```
                                    || gencode(一, Expression1.place, Term.place, Expression.place)
                                       }
Expression  →Term                   { Expression.place = Term.place;
                                      Expression.code = Term.code; }
Term  →Term1 '*' Factor             { Term.place = newtemp;
                                      Term.code = Term1.code
                                              || Factor.code
                                    || gencode( *, Term1.place, Factor.place, Term.place )
                                              }
Term  →Term1 '/' Factor             { Term.place = newtemp;
                                      Term.code = Term1.code
                                              || Factor.code
                                    || gencode( /, Term1.place, Factor.place, Term.place )
                                              }
Term  →Factor                       { Term.place = Factor.place;
                                      Term.code = Factor.code; }
Factor  →'('Expression')'           { Factor.place = Expression.place;
                                      Factor.code = Factor.code; }
Factor  →Indentifier                    { Factor.place = newtemp;
                                      p = lookup(Identifier.text);
                                      if( p!= NULL ){
                                              Factor.code = ' ';
                                              Factor.place = *p.place}
                                      else      ERROR();
                                      }
Factor  →Integer                    { Factor.place = newtemp;
                                    Factor.code = gencode( MOV, #Integer.text, Factor.place );
                                      }
```


4、 判断每个赋值语句右部是否是常量表达式的翻译方案：



```
Statement  →Identifier ':=' Expression   { Statement.true = newlable;
                                           Statement.false = newlable;
                                           Expression.true = Statement.true;
                                           Expression.false = Statement.false;
                                           Statement.code = Expression.code
                                                 || gencode(Statement.true':')
                                                 || gencode(print("Is a Integer Expression!"))
                                                 || gencode(Statement.false':')
                                                 || gencode(print("Is not a Integer Expression!"))
                                                 }
Expression  →Expression1 '+' Term      { Expression1.true = newlable;
Expression  →Expression1 '—'Term         Expression1.false = Expression.false
                                         Term.true = Expression.true;
                                         Term.false = Expression.false;
                                         Expression.code = Expression1.code
                                              || gencode( Expression1.true':')
                                              || Term.code
                                        }
Expression  →Term                      { Term.true = Expression.true;
                                         Term.false = Expression.false;
                                         Expression.code = Term.code;
                                         }
Term  →Term1 '*' Factor                { Term1.true = newlable;      Term1.false = Term.false;
Term  →Term1 '/' Factor                  Factor.true = Term.true;    Factor.false = Term.false;
                                         Term.code = Term1.code
                                              || gencode( Term1.true':' )
                                              || Factor.code
                                         }
Term  →Factor                          { Factor.true = Term.true;    Factor.false = Term.false;
                                         Term.code = Factor.code;
                                         }
Factor  →'(' Expression ')'            { Exression.true = Factor.true;
                                         Expression.false = Factor.false;
                                         Factor.code = Expression.code;
                                         }
Factor  →Indentifier                   { Factor.code = gencode( GOTO, __, __, Factor.false ) }
Factor  →Integer                       { Factor.code = gencode( GOTO, __, __, Factor.true ) }
```

三．1. 画出 NFA

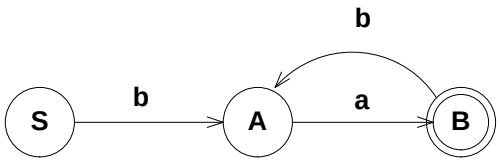


初始状态**S**，终止状态**B**

2. 构造状态转换表

	a	b
S	/	A
A	BC	/
BC	/	A

3. NFA 到 DFA



初始状态**S**，终止状态**B**

DFA 已经最小化

4. 写出相应的正则文法

S →bA　　或　　B ::= Aa

A →aB　　A ::= Bb | b

B →bA | ε

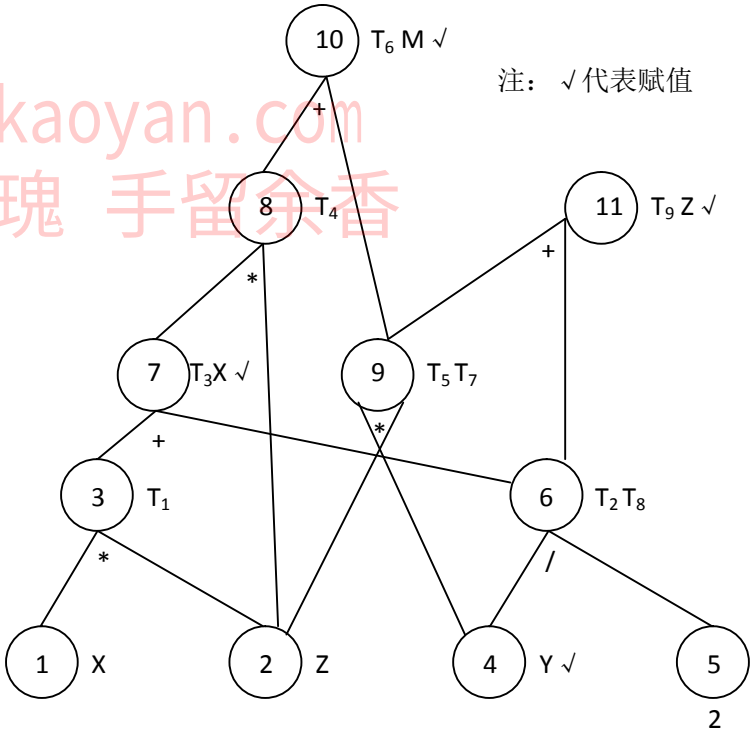
四. 1. 写出优化前的四元式：

(1) (*, X, Z, T_1)
(2) (/, Y, 2, T_2)
(3) (+, T_1, T_2, T_3)
(4) (=, T_3, __, X)
(5) (*, X, Z, T_4)
(6) (*, Y, Z, T_5)
(7) (+, T_4, T_5, T_6)
(8) (=, T_6, __, M)
(9) (*, Z, Y, T_7)
(10) (/, Y, 2, T_8)
(11) (+, T_7, T_8, T_9)
(12) (=, T_9, __, Z)

3. 优化后的四元式：

(1) (*, X, Z, T_1)
(2) (*, Y, 0.5, T_2)
(3) (+, T_1, T_2, T_3)
(4) (:=, T_3, __, X)
(5) (*, X, Z, T_4)
(6) (*, Z, Y, T_5)
(7) (+, T_4, T_5, T_6)
(8) (:=, T_6, __, M)
(9) (+, T_5, T_2, T_9)
(10) (:=, T_9, __, Z)

2. 画出 DAG 图：



注：√代表赋值

2007 年

【共 70 分】

一．有文法 G[E]如下：

E ::= E + T | E – T | T

T ::= T * F | F

F ::= (E) | i

其中 i 为整数。

1. 消除上述文法的左递归（5 分）

2. 用递归子程序法写出上述文法的识别程序（5 分）

3.假设 i 由词法分析程序给出，其值由 i.val 给出，试修改上述识别程序，使其能正确计算出表达式的值。（5 分）

二．对于文法 G[E]：

E ::= aA | bB

A ::= cA | d

B ::= cB | d

的增广方法 G'[Z]：

Z ::= E#

E ::= aA | bB

A ::= cA | d

B ::= cB | d

给出它的 LR(0)项集，并画出相应的特征状态机。（20 分）

三．给出 $L=\{a^n b^m c^k \mid m=n+k, n\geq 1, m\geq 1, k\geq 1\}$的文法描述。（5 分）

四．对于语言{{0}{1}}

1. 给出与之等价的 NFA（5 分）

2. 把上述 NFA 确定化成 DFA 并将其最小化（10 分）

五．指出下面这个流图中的可用表达式（可以不必写出数据流方程），并进行“消除公共子表达式”全局优化。（15 分）

（图略）

2007 年答案

一. 1. E ::= TE’

E’ ::= +TE’ | —TE’ | ε

T ::= FT’

T’ ::= *FT’ | ε

F ::= (E) | i

2. GetSymbol( ){ … };

Error( ){ … };

```
void E()
{   T();    E1();   }
void E1()
{
    if( sym == '+' || sym == '—' )
    {   GetSymbol();    T();    E1();   }
    else    return;
}
void T()
{   F();    T1();   }
void T1()
{
    if( sym == '*' )
    {   GetSymbol();    F();    T1();   }
    else    return;
}
void F()
{
    if( sym == '(' )
    {
        GetSymbol();    E();
        if( sym == ')' )
        {   GetSymbol();    return;   }
        else  Error();
    }
    else if( sym == ';' )
    {   GetSymbol();    return;   }
    else Error();
}
void main()
{   GetSymbol();    E();    }
```

3. S ::= E { print(E.val) }

E ::= E1 + T { E.val = E1.val + T.val }

E ::= E1—T { E.val = E1.val—T.val }

E ::= T　　{ E.val = T.val }

T ::= T1 * F　　{ E.val = T1.val * F.val }

T ::= F　　{ T.val = F.val }

F ::= (E)　　{ F.val = E.val }

F ::= i　　{ F.val = i.val }

二．增广型文法 G'[Z]为

Z ::= E #

E ::= aA

E ::= bB

A ::= cA

A ::= d

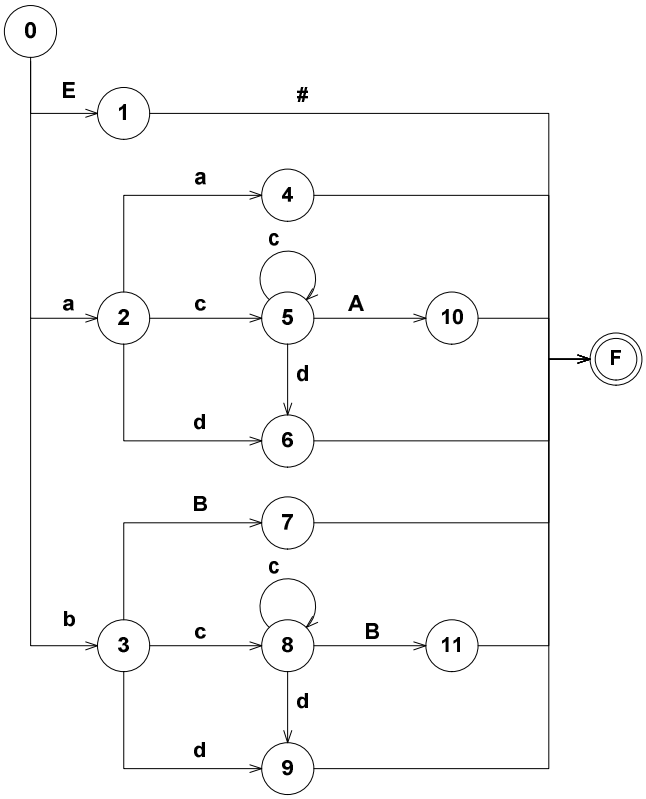
B ::= cB

B ::= d

1. LR(0)项目集簇：

状态	LR(0)项目集		后继
I_0	Z →.E# E →.aA E →.bB	E a b	I_1 I_2 I_3
I_1	Z →E.#	Z ::= E #	acc
I_2	E →a.A A →.cA A →.d	A c d	I_4 I_5 I_6
I_3	E →b.B B →.cB B →.d	B c d	I_7 I_8 I_9
I_4	E →aA.	E ::= aA	
I_5	A →c.A A →.cA A →.d	A c d	I_{10} I_5 I_6
I_6	A →d.	A ::= d	
I_7	E →bB.	E ::= bB	
I_8	B →c.B B →.cB B →.d	B c d	I_{11} I_8 I_9
I_9	B →d.	B ::= d	
I_{10}	A →cA.	A ::= cA	
I_{11}	B →cB.	B ::= cB	

2. 特征状态机 CFSM

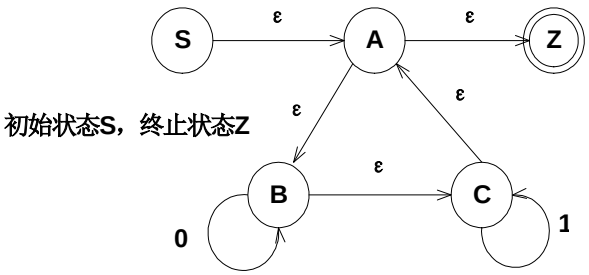


三．G[S]:　　S ::= AB

A ::= aAb | ab

B ::= bAc | bc

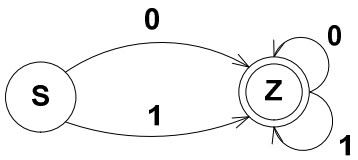
四．1. NFA 为：



初始状态S，终止状态Z

2. NFA 到 DFA

	0	1
SABCZ	ABCZ	ABCZ
ABCZ	ABCZ	ABCZ



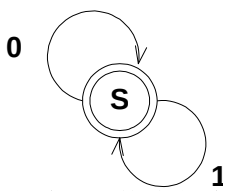
初始状态S，终止状态Z

NFA 最小化

A = { S, Z }　　$\{S\}_0 \subseteq A$　$\{S\}_1 \subseteq A$

$\{Z\}_0 \subseteq A$　$\{S\}_1 \subseteq A$

∴S 与 Z 状态等价，上图可最小化为：



初始状态S，终止状态S

五．参考 03-04 期末考试第四题。

2008 年

【试卷暂缺】

2008 年答案

【答案暂缺】

2009 年

一．写出所有字符由 a 或 b 构成，且 a 与 b 的个数相等的上下文无关文法。

二．已知一个 int 占用 4 个存储单元，bool 占用 2 个存储单元，写出下面文法的翻译方案，其中包括变量证明和变量大小。

Dec→TYPE D

D→id,D|id

TYPE→int|bool

其中可以使用 addIdentifer(id.lexval, id.type, address)把变量的值、类型和位置登记到符号表。

三．写出下列表达式的四元式，并说明循环体包含几个基本块，在循环体中有哪些循环不变量，是否可以将这些循环不变量外提。

```
int x, y, a, b, c;
x = a + b * c;
while(a < b)
{
    x = b * c;
    y = a + x;
    a = a + 1;
}
```

三．从字符串{ab}a{ab|ba}构造相应的 NFA，然后将 NFA 确定化并最小化。

五．文法 G(E)为：E→E*E

E→E+E

E→number

证明文法 G 为二义性文法，给出与文法 G 等价的非二义性文法，且+与*的优先级满足先加后乘。

2009 年答案

一．S::=AB

B::=bA|b

A::=aB|a

二．答案有疑问，顾不给出

三．提示：参照 2003-2004 年第二学期期末考试试卷

四．NFA⟶DFA⟶最小化

五．E::=E*T

E::=T

T::=T+F

T::=F

F::=i

2010 年

第一题：2003-2004 期末试题第二大题
第二题：2003-2004 期末试题第一大题 3 小题（加上抽象语法树）
第三题：2005-2006 期末试题第四大题
第四题：09 最后一道题，加上求 LR（0）规范簇，以及移入规约冲突

注：由于题目比较长，所以只记了题型，大家可以仿照后面的期末例题进行复习

2010 年答案

【答案暂缺】

2012 年

填空题：
有 4 个小题，都很简单，不记得了

解答题：

1. 若 LR(1)不含冲突项，合并同心集后 LALR(1)必不含何种冲突？为什么？

2. 对正则式 b{a|b}*构造 NFA，并用子集法将其确定化为 DFA（不需要最小化）

3. 为文法 S->repeat S1 while B 编写翻译方案。说明：S1 和 B 已经翻译，S1 有未确定的 nextlist 综合属性（S 也具有该属性），B 有未确定的 truelist 和 falselist 属性，除此之外不必考虑其他属性。可以添加 M、N 改变文法后再翻译，且只有 instr 属性，用于回填标号。必须使用 backpatch！不用的家伙不给分！

2013 年

填空题：

1. 编译有几个过程，回填属于哪个过程

2. 给出一个句子，用左推导等到该式子

3. 求 FIRST 集和 FOLLOW 集

解答题：

1. 能被 4 整除的个数不少于 2 的句子（如 100,0000），写出正则表达式和 DFA（要最小化）

2. 求 LR(1)的项集闭包（参考 03 年第三题）

3. 逆波兰表达式的翻译，将左递归改成非左递归的翻译（参考 03 年第四题）

2014 年

填空题：

1. 给出一个符号串集合，写出对应的产生式

2. 给出几个产生式，以及一个符号串，问句柄是什么？规约所用到的产生式是什么？

3. 三维数组求内存所在的地址

4. 数据流分析中的一些问题（向前流和向后流等等）

解答题：

1. 写出能被 4 整除的所有二进制数的正则表达式（000 属于表达，111000 属于表达式，1001 不属于表达式），求出 NFA 并化为 DFA 并最小化

2. S→xSyS|ySxS|空串
   - 要求根据给定文法写出长度小于 6 的全部串（6 个的不包跨在内）
   - （x(xy|yx)*y(xy|yx)*|y*(xy|yx)x(xy|yx)*）*，要求判断给定的正则表达式是否符合文法要求。若符合，回答是；否则，给出反例

3. 已知 S->while B do S1 的翻译方案，写出语句 S→do S1 while B 带回填的翻译方案

2003-2004 学年第二学期《编译原理》期末考试

一：填充题（如无特殊说明，每空两分）

1. 在标准的编译程序的组成部分中，属于前端部分的有：_____，_____，和_____。
2. 给出语言为 $\{a^n b^j c^i \mid n \geqslant 0,\ j \geqslant 0,\ 0 \leqslant i \leqslant n\}$ 对应的文法 G[Z]: _____。这个语言可以使用正则文法来描述吗？_____，如果能，给出正则文法，如果不能，简单说明理由：_____
_____。
3. 给定文法 G[E]: E→T+E | T　　T→F*T | F　　F→ ‘(’ E ‘)’ | i；给出句子 (i*i)的最左推导过程：_____，并给出语法树：

_____。
4. 给定文法：G[Z]: Z→WV　W→aB | aW | a　B→b | bB　V→bV | cC　C→c | cC。说明文法 G[Z]为二义性的：

_____。（本空 4 分）
给出文法的语言：_____。
并给出等价的无二义性的文法：
_____。（本空 4 分）
5. 给出和正则文法 W→A0　A→A0 | W1 | 0 等价的有穷状态自动机：

_____。（本空 4 分）
6. 对于赋值语句序列 c = (a+b)*(c+d)；x = (a+b)-(c+d)。给出对应的四元式序列：

_____。
找出公共子表达式并进行优化后得到如下四元式序列：

_____。(本空 4 分)

从上面的赋值语句序列中得到等价的逆波兰表达式：

_____。

7. 对于文法 G[S]: S→ a | b | (T) T→T,S | S ；给出 LR(0)项集{T→ · T,S}的闭包：

_____。

8. 给定下列文法 G[E]以及翻译方案

E→T+E_1 {E.val = T.val + E1.val}

E→T–E_1 {E.val = T.val-E1.val}

E→T {E.val = T.val}

T→ i {T.val = i.lexVal}

按照这个翻译方案，求解句子 9-3-2 的值：

_____。

9. 给出(a+b)*(c-d)+e 的抽象语法树：

_____。

10. 简单解释为什么 SLR(1)分析技术的处理范围要比 LR(1)技术的处理范围小？

_____。

11. 说明在程序运行的时候下列变量/值的内存分配方式：(1)局部变量 vlocal_____ (2)全局变量 p 指向的整数值，该值由 malloc 申请空间_____。(静态分配填写 a，栈式分配填写 b，堆式分配填写 c)。

二：使用递归子程序方法扫描文法 G[P]的句子，并作语义处理。(20 分，其中语法扫描 12 分，语义处理 8 分)

G[P] P→D ';' S '.' D → T id ';' D | 空 T → int | bool

S → Assign ';' S | 空 Assign → id ':=' E

E → id + id | id and id

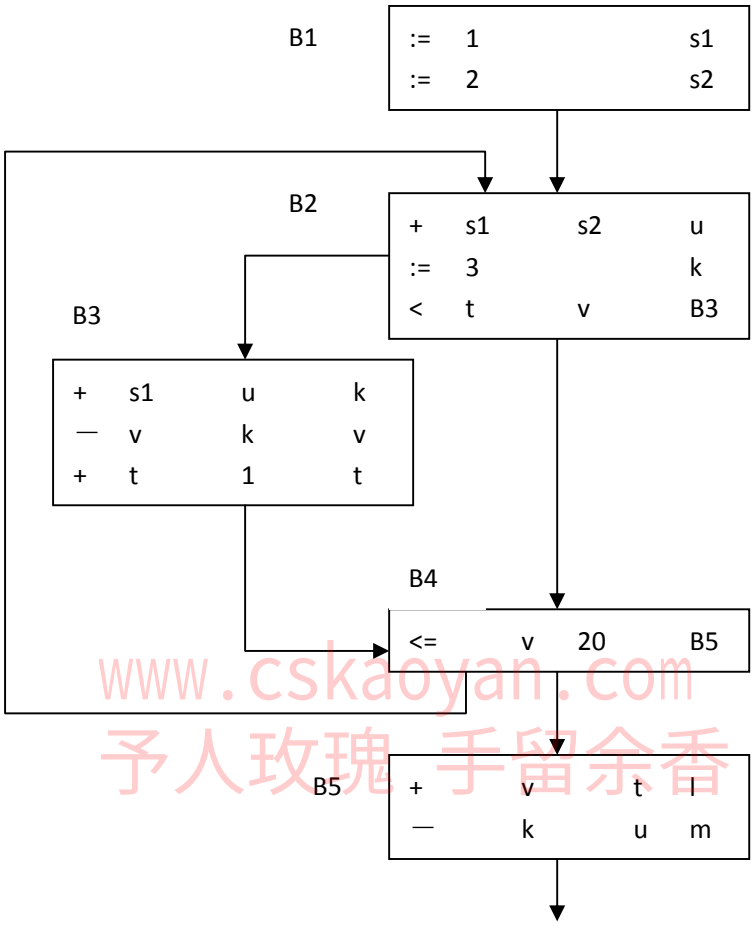
语义处理的要求如下，在句子的申明部分(D)，说明了一些变量，变量的类型可以是 int 或者 bool 型。语义处理时，必须将变量记录到标识符表中。在处理语句部分的时候进行类型检查：首先+号的两端必须是整数型变量，and 的两端必须是 bool 型；其次，赋值的时候，两端的类型必须相同。

（使用 AddIdEntry(id.lexVal, type)将 id 的类型信息存放到标识符表，使用 GetIdType(id.lexVal)获取标识符的类型。在扫描程序碰到语法/语义错误的时候，可以立刻中止运行，不必作错误恢复。）

三：为下列正则表达式构造 NFA，并进行确定化和最小化。

1{10}{10 | 01} (15 分，构造 NFA 5 分,确定化 5 分,最小化 5 分)

四：下面的流图中，循环为 B2,B3,B4 组成了一个自然循环。找出该循环中所有不变四元式（5 分，不必写出数据流方程）；确定哪些不变四元式不可以外提并简单说明理由（5 分）；将不变表达式外提，形成优化后的流图(5 分)



2004-2005 学年第二学期《编译原理》期末考试

一、填充题（如无特殊说明，每空 2 分）

1、编译程序的前端部分包括：词法分析，_____，_____三个部分。

2、请列出 2 种在基本块内部的优化方法：_____，_____。

3、在 C 语言程序中，局部变量 int p 存放的位置是_____；语句 p=malloc(sizeof(int)10)申请得到的空间位于_____；全局变量 int globalIndex 存放的位置是_____；局部变量 static int si 的存放位置是_____。（以上四个空请填写静态区，栈区，和堆区）

4、给出表达式 a+b*(c+d)的逆波兰表示：_____。

5、说明为什么文法 G[Z]: Z→AC A→aAc | Ac | cc | ac C→cB|cCB→bB|b 是二义性的_____

_____。并给出等价的无二义性的文法：_____

_____（本空 4 分）。

6、给定文法 G[Z]: Z→aZb | AB A→a | aAB→bB | b，该文法的语言是：_____

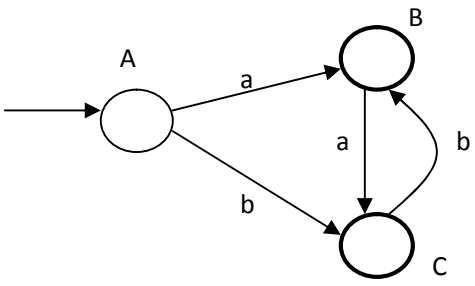
_____。该语言能否使用正则文法描述？_____。如果能，给出相应的正则文法，如不能，给出理由_____

_____。

7、给定文法 G[E]: E→T+E | T-E | T, T→number，请给出句子 2+3- 4 的最左推导过程（假设 2,3,4 都是 number）_____。

最右推导过程_____。给出语法树(本空 4 分)：

9、给定下图中表示的有穷自动机，请给出相应的正则文法_____。和正则表达式_____。



初始状态 A，接受状态为 B,C

10、对于下面的算法四元式序列，指出公共子表达式的四元式序号：_____。

并在右边给出针对公共子表达式的优化之后的四元式序列(假设 t1,t2,t3,t4 不在基本块外使用）。

(1) + x y t1
(2) * t1 z t2
(3) + x y t3
(4) + t2 t3 y
(5) + x y t4
(6) * t3 t4 m _____。

11、给定文法 G[S]的部分规则: S→ v := E | if B then S else fi | if B then S fi | while B then S end B→B or v |B and v | v 请给出 LR(1)项集{[S→if then • S else fi, #]}的闭包 _____。其中大写字母为非终结符号，带下划线的小写字母符号串代表终结符号。该闭包的 if 后继项集闭包是 _____。

12、已知文法 G[Z]：Z→APZ | AMZ | AMB A→a | aA P→ +P|-P | - | + M→*M B→b|bB，请对文法进行压缩变换，得到新的压缩了的文法如下：_____。（本空 4 分）

13、给定正则表达式 a{b}c，给出相应的 DFA（本空 6 分）

_____。

二、已知用逆波兰方法表示的四则运算表达式的文法如下

G[E] E→ E E B | E U | number B→ < | > | + | - | and | or
U→ minus | ord

其语义解释如下：

<, > 是比较操作，其操作数必须是整数型，结果为 boolean 型，
+, - 为加法和减法操作，其操作数为整数型，结果为整数型。
and, or 为 boolean 型，其操作数和结果都是 boolean 型。
minus 为求负操作，其操作数和结果都是 boolean 型
ord 为序数操作，其操作数为 boolean 型，结果为 int 型。

请给出进行类型检查的翻译方案。(15 分）

三、请给出 do-while 语句的代码生成的模板，以及相应的翻译方案。已知其语法如下：

S → do S while E

其中假设其他类型的语句，和表达式的翻译方案已经生成。（15 分，如果不使用回填技术扣 5 分）

四、给出下列文法的 LL(1)分析表。（小写字符串表示终结符号）（10 分）

E→ -E | (E) | Var Etail Etail → lembda | -E Var → id Vtail
Vtail→(E) | lambda

2005-2006 年度第二学期《编译原理》期末考试

一、填充题（如无特殊说明，每空 2 分）

1、 编译程序可以分成以下几个部分：_____，语法分析，_____，代码生成，和优化。

2、 针对循环的优化技术包括：_____，_____。（写出两种即可）

3、 说明四则运算表达式的文法 E→E+E ， E→E*E ，E→number 是二义性的文法（3 分）。

_____

_____。并给出等价的无二义性的文法，其中的 + 和 * 的优先级为**先加后乘**（3 分）：_____

_____。

4、 已知正则表达式 a{b|c}d，给出相应的 DFA：（4 分）

_____

以及相应的正则文法：（2 分）

_____

_____。

5、给定下列左递归文法： T→T*F | F F→i | (E)。请给出等价的无左递归的文法（4 分）：

_____。

a) 设有文法 G[S]: S→a | b | (T) T→ T,S | S （其中(和)为终结符号）。给出对应于规则 T→T,S 的全部 LR(0)项：_____。给出本文法的 LR(1)项集 {[T→T, ● S, ,]} 的闭包：_____

_____。

b) 给定如下的四元式序列，请在右边给出消除公共子表达式之后的四元式序列(4 分)：

```
*  x  y  t1
+  t1 z  t2
*  x  y  t3
*  t2 t3 t4
*  x  y  t5
+  t5 z  x
*  x  y  m
*  m  t4 x
```

_____

8、给出语言为{$a^n b^j c^i$ | n≥0， j≥0，0≤i≤j}的文法 G[Z]:_____

_____。这个语言可以使用正则文法来描述吗？_____。

9、给出四则运算表达式(a+b)*(c+d)/a-b 的抽象语法树（4 分）：

_____。并给出这个表达式的逆波兰表示方式：_____

_____。

10、是否存在能用 SLR(1)技术的处理，但是不能被 LALR(1)技术处理的文法？

_____。

11、已知文法 G[Z]：Z→APZ | AMZ | AMB | AP A→a | aA P→ +P|-P | - | + M→*M

B→b|bB，进行压缩变换后的文法如下：（4 分）_____
_____。

12、在 C++语言中，static 的类变量的存放位置是：_____。假设有类 ClassA，而 memVar 是 Class 的普通成员变量。假设在程序中有代码 Class * obj=new Class，那么 obj→memVar 的存放位置是：_____；假设程序中使用 ClassA gblobj 的方式声明全局变量，那么 gblobj.memVar 的存放位置是：_____。（以上三个空格填写堆区，栈区和静态区）

13、写出赋值表达式 x = a*b-c+d*e 的四元式序列（4 分）

_____

和相应的三元式序列：

_____。

二、设有文法 G[E]: E→T+E | T-E | T　　T→ F*T | F　　F→(E) | i

1、请使用递归下降分析技术给出扫描该文法句子的程序。必要时先修改文法。（10 分）

2、修改上面的程序，使之能够统计扫描的句子中总共出现了多少对括号。（5 分）

三、已知有选择表达式的文法如下：Exp→ (BoolExp ? Exp : Exp); 其语义如下：首先对 BoolExp 求值，如果值为真则对第一个 Exp 进行求值，其结果为第一个 Exp 的值，如果 BoolExp 的值为假，则对第二个 Exp 求值，表达式的最终结果为第二个 Exp 的值。请给出生成相应代码的翻译方案。（10 分，如果不使用 BackPatch 扣 3 分）

四、已知源程序如下：

```
int x,y,a,b,c;
x = a+b*c;
while(a < b)
{
    x = b*c;
    y = x + a;
    a = a + 1;
}
```

1、给出相应的四元式序列（5 分）。

2、给出相应的流图（5 分）。

3、指出其中的循环不变表达式，并外提（5 分）。