清华大学本科生考试试题专用纸

考试课程 《编译原理》 （A 卷） 2012 年 1 月 5 日

学号：_____ 姓名：_____ 班级：_____

（注：解答可以写在答题纸上，也可以写在试卷上；交卷时二者都需要上交。）

一.（15分）

右图是某简单语言的一段代码。该语言支持嵌套的过程声明，但只能定义无参过程，且没有返回值。语言中不包含数据类型的声明，所有变量的类型默认为整型。语句块的括号为‘begin’和‘end’组合；赋值号为 ‘:=’。每一个过程声明对应一个静态作用域，最外层的作用域编号为第0层，依次类推。我们假设在实现该语言是采用多符号表结构，每个静态作用域对应一个符号表，且通过单独的一遍建立符号表（类似本学期实验框架，这样，当前用到一个符号不一定要求前面已声明过）。如，右图程序中第0层作用域包含符号 a, b, P, Q。不同的作用域包含不同的符号，但可以重名，并遵守一般的可见性原则。

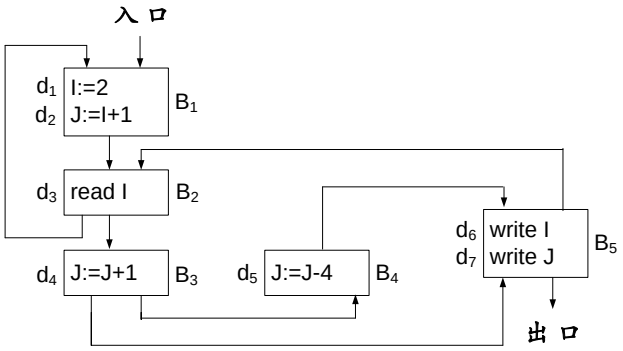
```
(1)  var a,b ;
(2)  procedure  P ;
(3)       var x;
(4)        procedure  R ;
(5)            var x, a;
(6)            begin
(7)                a := 3;
(8)                call Q;
                   …… /*仅含符号 x*/
.              end;
.          begin
.              call R;
.              …… /*仅含符号 x*/
.          end ;
.     procedure  Q ;
.          var x;
.          begin
(L)            if  a < b  then call P ;
.              …… /*仅含符号 x*/
.          end ;
.     begin
.          a := 1;
.          b := 2;
.          call Q;
.          ……
.     end .
```

1.（6分）编译器使用一个作用域栈来记录当前的开作用域。对于右上图的程序代码，当编译器在处理到第8行时作用域栈中有哪些作用域？按照从栈底到栈顶的次序依次指出这些开作用域，并列出每个作用域对应的符号表中所有的符号。

2.（5分）假设采用 Display 表机制来实现对非本地局部变量的访问。当右上图的程序第二次执行到第 L 行（还未执行）时，Display 表的各个分量保存的是哪个过程第几次被激活时的过程活动记录在运行栈中的基址？（提示：对于该程序，Display 表有三个有效分量，分别用下标 0、1、和 2 对应）

3.（4分）针对静态作用域规则和动态作用域规则，该程序的执行效果有何不同？

二 (10分) 下图是包含 5 个基本块的流图（为方便，省略了基本块内的转移语句），其中 B1 为入口基本块：



1. (3分) 指出在该流图中，基本块 B_5 的支配结点（基本块）集合，始于 B_5 的回边，以及基于该回边的自然循环中包含哪些基本块?

2. (3分) 指出基本块 B_2 入口处和出口处的活跃变量集合（假设基本块 B_5 的出口处活跃变量集合为空集）。

3. (2分) 指出该流图范围内，变量 J 在 d_7 的 UD 链。

4. (2分) 指出该流图范围内，变量 J 在 d_5 的 DU 链。

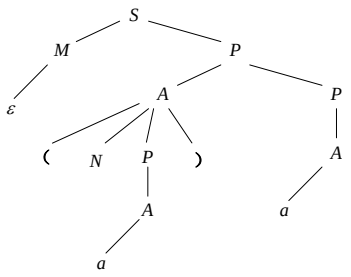
三 **(15 分)** 给定 SLR(1) 文法 $G[S]$:

$$S \to M\ P$$
$$P \to A\ P \mid A$$
$$A \to (\ P\ ) \mid a$$
$$M \to \varepsilon$$

如下是基于G[S]的一个经过调整使得继承属性可模拟的 L 翻译模式:

$S \to M$ { $P.y := M.x$ } P { $S.x := P.x$ } /* := 为赋值号*/
$P \to$ { $A.y := P.y$ } A { $P_1.y := A.x$ } P_1 { $P.x := A.x + P_1.x$ }
$P \to$ { $A.y := P.y$ } A { $P.x := A.x$ }
$A \to$ ({ $N.y := A.y$ } N { $P.y := N.x$ } P) { $A.x := P.x$ }
$A \to a$ { $A.x := A.y + 1$ }
$M \to \varepsilon$ { $M.x := 0$ }
$N \to \varepsilon$ { $N.x := N.y$ }

1. (3分) 输入串 (a) a 的一棵语法分析树如下图所示。通过对该语法分析树进行标注，可得到一棵带标注的语法分析树。试指出在这一带标注的语法分析树中根节点对应的文法符号的属性值，即 $S.x$ = ？



2. (6分) 试分别指出句型 M（P）A 和 M（A）a 的所有短语，直接短语。如果这些句型同时也是右句型，那么还要给出其句柄. 请将结果填入下表中:

句型	短语	直接短语	句柄
$M(P)A$			
$M(A)a$			

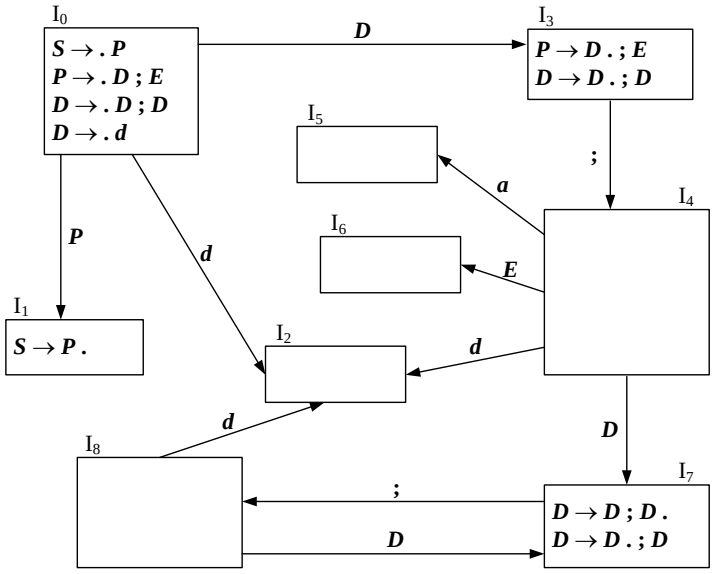
3. (6分) 如果在 LR 分析过程中根据该翻译模式进行自下而上翻译，试写出在按每个产生式归约时语义处理的一个代码片断（设语义栈由向量 v 表示，归约前栈顶位置为 top，终结符不对应语义值，而每个非终结符的综合属性都只对应一个语义值，可用 v[i].x 表示；不用考虑对 top 的维护）。

四 (24分) 给定如下文法 $G(P)$：

(1) $P \to D\ ;\ E$
(2) $D \to D\ ;\ D$
(3) $D \to d$
(4) $E \to a$

为文法 $G(P)$ 增加产生式 $S \to P$，得到增广文法$G'[S]$。

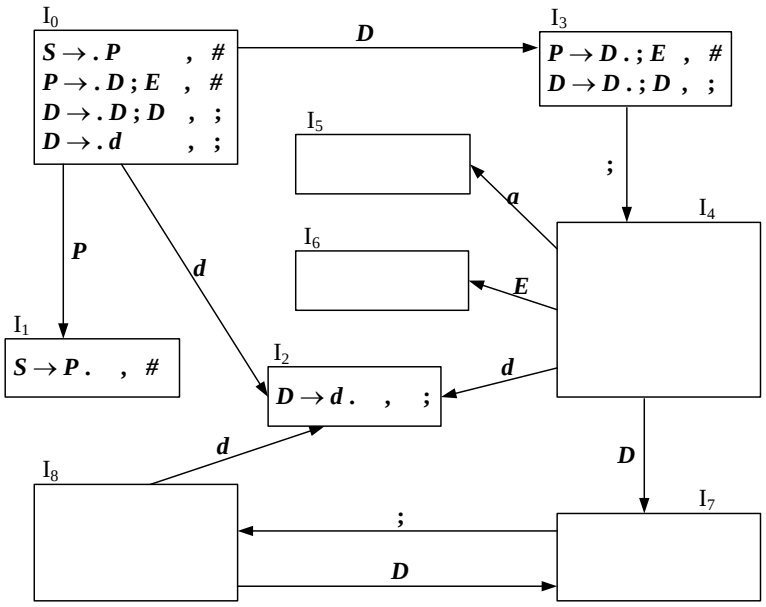
1. (5分) 下图表示该文法的LR(0)自动机，部分状态所对应的项目集未给出，试补齐之（即分别给出状态 I_2，I_4，I_5 ，I_6，和 I_8 对应的项目集。



2. (2分) 指出上图中LR(0)自动机中所有冲突的状态（并指出是哪种类型的冲突），以说明该文法不是 LR(0) 文法。

3. (2分) 针对所有冲突的状态，说明这些冲突不适合采用 SLR(1) 分析方法解决。

4. (5分) 下图表示该文法的LR(1)自动机，部分状态所对应的项目集未给出，试补齐之（即分别给出状态 I_4，I_5，I_6 ，I_7，和 I_8 对应的项目集。



5. (2分) 指出上图中LR(1)自动机中所有冲突的状态（并指出是哪种类型的冲突），以说明该文法不是 LR(1) 文法。

6. (6分) 尽管该文法不是LR文法，但仍有可能采用LR分析方法完成语法分析。（1）试给出一种采用SLR(1)分析方法的解决方案；（2）根据你所给的方案完成下图的 SLR(1) 分析表，状态 4、5、7 和8 对应的行未给出，试补齐之 （若你的方案与下图不匹配，请给出相应的完整SLR(1) 分析表）。

状态	ACTION				GOTO		
	a	*d*	;	#	*P*	*D*	*E*
0		s_2			1	3	
1				acc			
2			r_3				
3			s_4				
4							
5							
6				r_1			
7							
8							

7. (2分) 变换文法，消除文法 *G*(*P*)中的左递归。

五 (20 分) 给定文法 *G*[*S*]：

$$S \to P \mid \varepsilon$$
$$P \to (\ P\ )\ P \mid a$$

1. (5分) 针对文法 *G*[*S*]，试给出各产生式右部文法符号串的 *First* 集合，各产生式左部非终结符的 *Follow* 集合，以及各产生式的预测集合*PS*，即完成下表：

G 中的规则 *r*	*First* (*rhs*(*r*))	*Follow* (*lhs*(*r*))	*PS* (*r*)
$S \to P$			
$S \to \varepsilon$		此处不填	
$P \to (P)\ P$			
$P \to a$		此处不填	

表中的 *rhs(r)* 表示产生式 *r* 右部的文法符号串，*lhs(r)* 表示产生式 *r* 左部的非终结符。

2. (2分) 文法 *G*[*S*] 是 *LL*(1) 文法，请说出为什么？

3. (4分) 给出该文法的预测分析表，即完成下表：

	a	(	)	#
S				
P				

4. (9分) 如下是以 LL(1) 文法 *G*[*S*] 作为基础文法设计的翻译模式：

$S \to \{\ P.y := 0\ \}\ P\ \{\ S.x := P.x\ \}$ /* := 为赋值号*/
$S \to \varepsilon\ \ \{\ S.x := 0\ \}$
$P \to (\ \{\ P_1.y := P.y\ \}\ P_1\ )\ \ \{\ P_2.y := P_1.x\ \}\ P_2\ \{\ P.x := P_1.x + P_2.x\ \}$
$P \to a\ \ \{\ P.x := P.y + 1\ \}$

试针对该翻译模式构造一个自上而下的递归下降（预测）翻译程序（下一个输入符号保存于全局量 lookahead, 可以直接使用讲稿中的 MatchToken 函数）。

注意：‘(’, ‘)’, 以及 ‘*a*’ 是基础文法的终结符。

六 (16分) 以下是语法制导生成某类TAC语句的一个 L-属性文法片段（类似讲稿中相应内容）：

$S \to$ *if E then* S_1
{ *E .true* := *newlabel ;*
E .false := *S .next ;*
S_1*.next* := *S .next ;*
S .code := *E .code* || *gen(E.true* ‘:’*)* || S_1 *.code*
}

$S \to$ *if E then* S_1 *else* S_2
{ *E .true* := *newlabel;*
E .false := *newlabel;*
S_1 *.next* := *S .next ;*
S_2 *.next* := *S .next ;*
S .code := *E .code* || *gen(E.true* ‘:’*)* || S_1 *.code* || *gen(‘goto’S.next)* || *gen(E .false ‘:’)*
|| S_2 *.code*
}

$S \to$ *while E do* S_1
{ *E .true* := *newlabel ;*
E .false := S .next ;
S_1 *.next := newlabel ;*
S .code := gen(S_1 *.next ‘:’)*|| *E .code*|| *gen(E.true* ‘:’*)* || S_1 *.code* || *gen(‘goto’*S_1 *.next)*
}

$S \rightarrow S_1; S_2$
{ S_1 .next := newlabel ;
S_2 .next := S .next ;
S .code := S_1 .code || gen(S_1 .next ':') || S_2 .code
}

$S \rightarrow S'$
{
S'.next := S .next ;
S .code := S'.code
}

$S' \rightarrow$ id := A
{ p := lookup (id.name);
S'. code := A.code || gen (p ':=' A . place)
}

$E \rightarrow E_1$ or E_2
{ E_1 .true := E . true ;
E_1 . false := newlabel ;
E_2 . true := E . true ;
E_2 . false := E . false ;
E .code := E_1 .code || gen(E_1 . false ':') || E_2 .code
}

$E \rightarrow id_1$ rop id_2
{ E.code := gen ('if' id1.place rop.op id2.place 'goto' E.true)
|| gen ('goto' E.false)
}

..... /*这里略去关于布尔表达式更多的部分*/

$A \rightarrow A_1 + A_2$
{ A .place := newtemp;

A .code := A_1 .code || A_2 .code || gen(A .place ':=' A_1 . place + A_2 . place)
}

..... /*这里略去关于算术表达式更多的部分*/

其中，属性 S .code , E .code , S .next , E.true , E.false，A .place, A . code,语义函数 newlabel , gen( ) 以及所涉及到的TAC语句与讲稿中一致，“||”表示TAC语句序列的拼接。

另外，要注意到本题中使用的文法非终结符含义：S（所有语句），S'（赋值语句），E（布尔表达式），A（ 算术表达式）。

（此外，假设在语法制导处理过程中遇到的冲突问题均可以按照某种原则处理比如规定优先级，else 匹配之前最近的 if，运算的结合性，等等，这里不必考虑基础文法是否 LR 文法。）

下面叙述本题的要求：

1. (8分) 设有开关语句

switch A of
case d_1 : S_1 ;
case d_2 : S_2 ;
.....
case d_n : S_n ;
default S_{n+1}
end

假设其具有如下执行语义：

(1) 对算术表达式 A 进行求值；

(2) 若 A 的取值为 d_1，则执行 S_1，转 (3)；

否则，若 A 的取值为 d_2，则执行 S_2，转 (3)；

.....

否则，若 A 的取值为 d_n，则执行 S_n，转 (3)；

否则，执行 S_{n+1}，转 (3)；

(3) 结束该开关语句的执行。

若在基础文法中增加关于开关语句的下列产生式

$S \rightarrow$ *switch A of L end*

$L \rightarrow$ *case V* : S ; L

$L \rightarrow$ *default S*

$V \rightarrow d$

其中，终结符 d 代表常量，其属性值可由词法分析得到，以 *d.lexval* 表示。

试参考上述 L-属性文法，给出相应的语义处理部分（不改变 L-属性文法的特征）。

注：可设计增加新的属性，必要时给出解释。

2. (8分) 若在基础文法中增加关于 *break* 语句的产生式

$S \rightarrow$ *break*

这里，*break* 语句的执行语义跳出直接包含该 *break* 语句的 *while* 语句。

试给出针对上述 L-属性文法的一种修改方案（不改变 L-属性文法的特征）。根据你的方案，给出如下产生式的语义处理部分：

$S \rightarrow$ *while E do S*

$S \rightarrow S ; S$

$S \rightarrow$ *break*

注：可设计增加新的属性，必要时给出解释。