中山大学软件学院 **2007，2008** 级软件工程专业

《编译原理》期末考试试题 A 卷答案

(考试形式： 闭卷 考试时间：2 小时)

警示

《中山大学授予学士学位工作细则》第六条

考试作弊不授予学士学位

方向：_____ 姓名：_____ 学号：_____

1. **(9 points) Answer the following questions.**

   (1) **How many phases does a compiler have? What are they?**

   编译程序分为：词法分析、语法分析、语义分析、中间代码生成、代码优化、代码生成、符号表管理和出错处理八个阶段.

   (2) **Write three names of parsing methods.**

   非递归的预测分析(LL(1))技术，算符优先分析技术，SLR 分析技术. (本题的答案不是唯一的.)

   (3) **What is a synthesized attribute? What is an inherited attribute?**

   在语法制导定义中，每个文法有一组属性. 设对产生式 A→α 有形式为 b := f(c1,c2,…,ck)的语义规则，其中 f 是函数，b 和 c1, c2, …, ck 是该产生式的文法符号的属性，并且

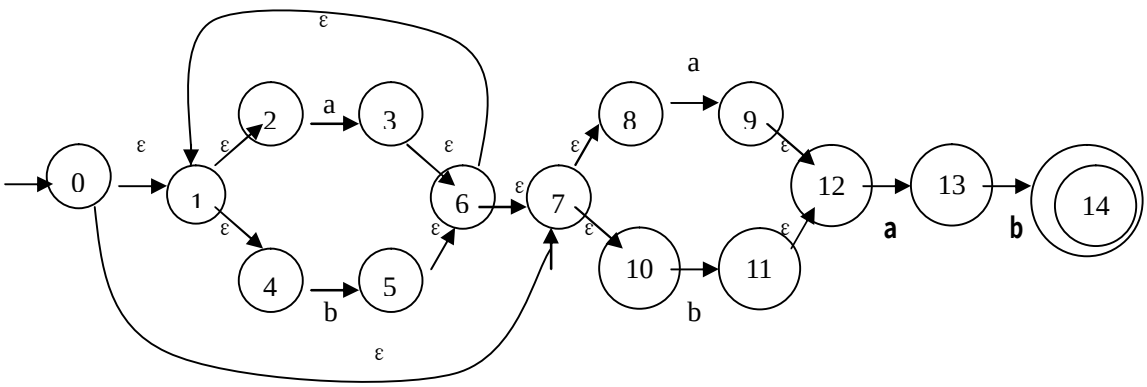
   (1)如果 b 是 A 的属性，c1,c2, …, ck 是产生式右部文法符号的属性或 A 的其它属性，那么,b 叫做文法符号 A 的综合属性.

   (2)如果 b 是产生式右部某个文法符号 X 的属性，c1,c2,…,ck 是 A 的属性,或右部文法符号的属性，那么,b 叫做文法符号 X 的继承属性.

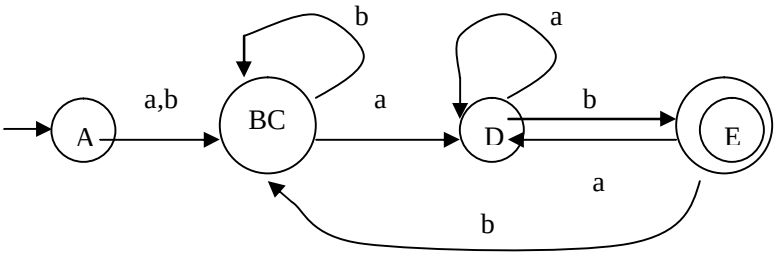
2. **(15 points) Consider the following regular expression:**

   (a | b)*(a | b)ab

   (1) **Convert the above regular expression into an equivalent NFA by Thompson Algorithm.**



   (2) **Construct an equivalent DFA with as few states as possible for the above regular expression.**



3. **(10 points) Are these following languages regular languages, context-free languages or non-context-free languages? Construct a regular expression for each regular language, and a context-free grammar for each context-free language.**

   (1) **L_1={w2w | w∈{0,1}* }.**

   不是上下文无关语言，是上下文有关语言.

   (2) **L_2={w | w∈{0,1}* and w does not contain the substring 101}.**

   是正则语言. 正则表达式为: **0*(1+00+)*1*0***

   (3) **L_3={w2w^R | w∈{0,1}* }. Note that w^R denotes the reverse of w. For example, 0121^R=1210.**

   是上下文无关语言. 上下文无关文法为: **S à 0S0 | 1S1 | 2**

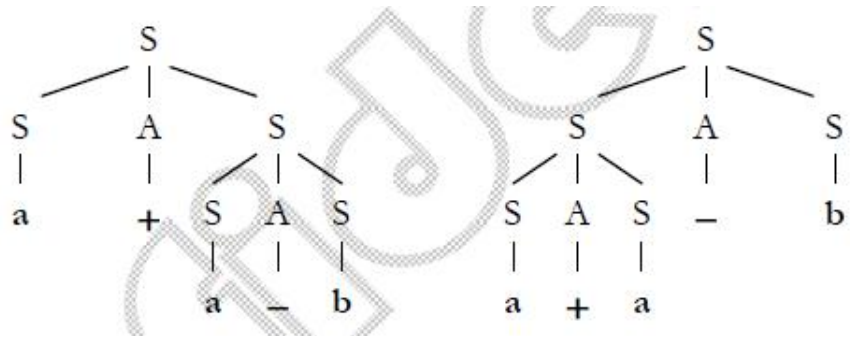
4. **(9 points) Consider the following grammar:**

   S à SAS | (S) | a | b

   A à + | -

   (1) **Demonstrate that this grammar is ambiguous.**

   对于句子 **a + a – b**，有两棵不同的分析树



   因而文法是二义文法.

   (2) **Create an unambiguous grammar that generates the same language as the grammar above.**

   S à SAT | T

   T à (S) | a | b

   A à + | -

5. **(13 points) Consider the following grammar over the alphabet {c, d, e}:**

   S à ABA

   A à Bc | dA | e

   B à eA

   (1) **Compute the FIRST and FOLLOW sets for each non-terminal in this grammar.**

符号	FIRST 集	FOLLOW 集
S	d e	$
A	d e ε	c d e $
B	e	c d e $

(2) Fill in the row corresponding to A in the LL(1) parsing table for this grammar.

	c	d	e	$
A	(a)	(b)	(c)	(d)

(a) A ® e

(b) A®dA, A®e

(c) A®Bc, A®e

(d) A®e

(3) Is this grammar LL(1)? Explain briefly why or why not.

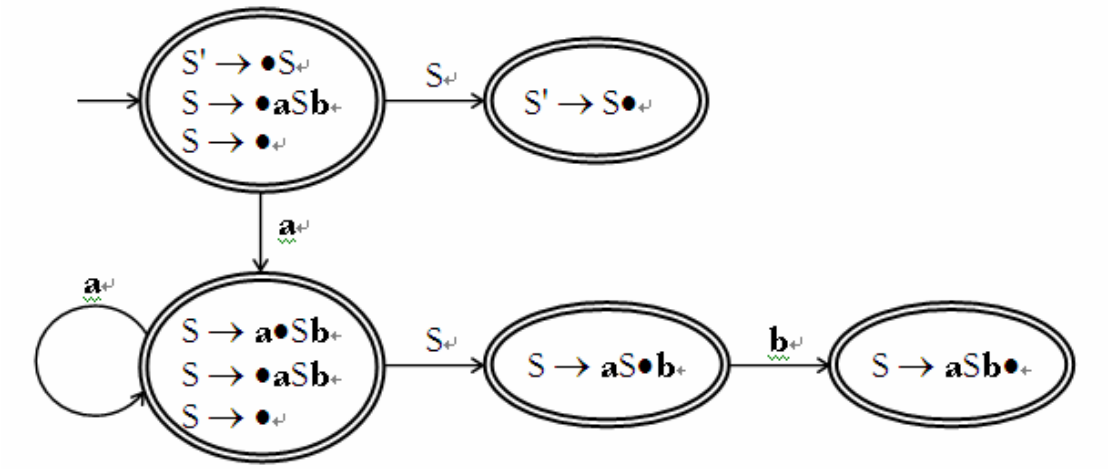
该文法不是 LL(1)的，因为它的分析表中至少有一个入口有多个产生式.

6. (10 points) Construct an LR(0) automaton for the following grammar, and determine whether it is an SLR(1) grammar.

S' ® S

S ® aSb | e

Note that the grammar is augmented, so you do not need to introduce any new start symbol.



由于仅在两个多项目的状态中才有移进一归约冲突，而 FOLLOW(S) = { b, $}，故可解析冲突，从而该文法是 SLR(1)的.

7. (12 points) The LR parsing table for an unknown grammar is given as follows:

	b	A	$	S	A
0	*s*3	*s*2		1	5
1			*accept*		
2	*r*4	*s*2	*r*4		4
3			*r*1		
4	*r*3		*r*3		
5	*s*6				

6		*s*2			7
7			*r*2		

The grammar is known to have the following productions:

(r1) S à ____(1)____

(r2) S à ____(2)____

(r3) A à ____(3)____

(r4) A à ____(4)____

Please fill in the blank (1)~(4) to complete the grammar according to the LR table above.

(1) b

(2) AbA

(3) aA

(4) a

8. **(10 points) Design a Syntax-Directed Translation scheme (SDT) for the following expression grammar, such that the non-terminal E has an attribute E.type which keeps the type of E. The value of E.type is BOOL, INTEGER or ERROR, where BOOL denotes the boolean type, INTEGER denotes the integer type, and ERROR denotes an illegal type.**

E à E_1 + E_2	**{**	**(1)**	**}**
| E_1 and E_2	**{**	**(2)**	**}**
| E_1 = E_2	**{**	**(3)**	**}**
| (E_1)	**{**	**(4)**	**}**
| true	**{**	**(5)**	**}**
| false	**{**	**(6)**	**}**
| int	**{**	**(7)**	**}**

Please fill semantic actions in the blank (1) ~ (7).

Note: The operator '+' must be applied to two integer expressions; the operator 'and' must be applied to two boolean expressions; the relational operator '=' must be applied to either two integer expressions or two boolean expressions, but cannot be applied to an integer expression and a boolean expression; if any operand is of illegal type, the resulting type is also illegal.

```
(1) E.type := ((E1.type = INT) and (E2.type = INT) ? INT : ERROR);
(2) E.type := ((E1.type = BOOL) and (E2.type = BOOL) ? BOOL : ERROR);
(3) E.type := ((E1.type = E2.type) and (E2.type ¹ ERROR) ? BOOL :
ERROR);
(4) E.type := E1.type;
(5) E.type := BOOL;
(6) E.type := BOOL;
(7) E.type := INT;
```

9. **(12 points) Consider the following basic block:**

(1) T0 := 3.14 **(7) B := A**

(2) T1 := 2*T0 **(8) T5 := 2*T0**

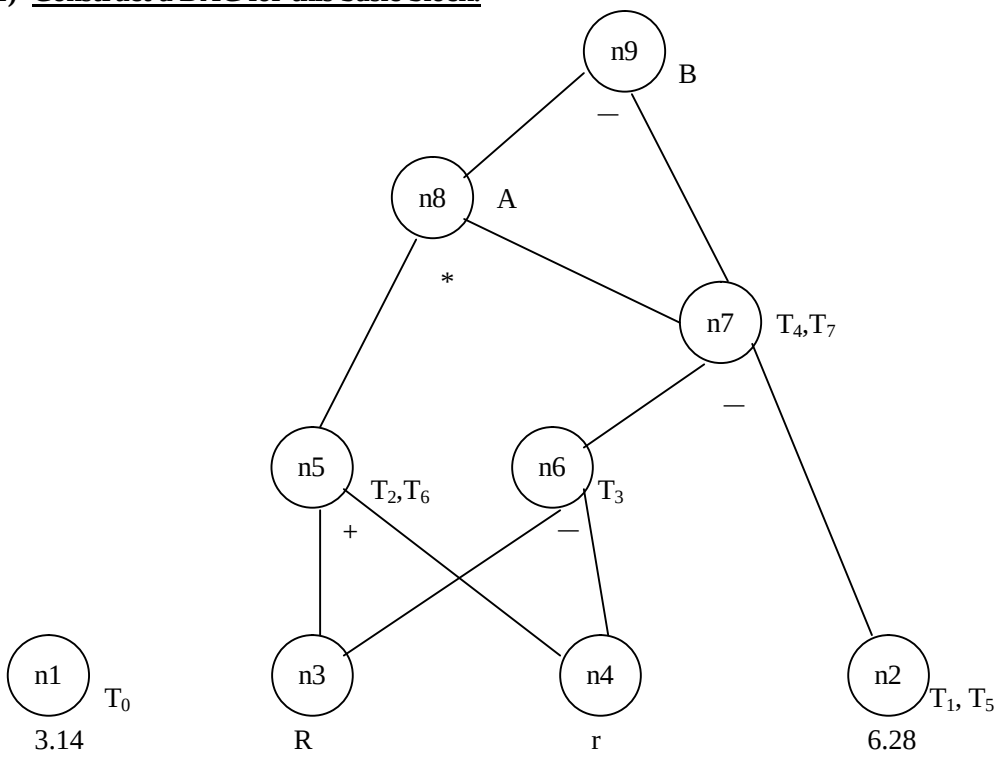
(3) T2 := R+r **(9) T6 := R+r**

(4) T3 := R－r **(10) T7 := T3－T5**

(5) T4 := T3－T1 **(11) B := A－T7**

(6) A := T2*T4

(1) <u>Construct a DAG for this basic block.</u>



(2) <u>Assuming that only A and B are live on exit from this basic block, simplify the three-address code.</u>

S1 := R+r

S2 := R－r

S3 := S2－6.28

A := S1*S3

B := A－S3

注意本题中的临时变量 S1~S3 可以替换为其它名称.