# 计算机与控制工程学院 2017－2018 学年第一学期
# 本科生编译系统原理期末考试试卷(A 卷)

专业：_____年级：_____学号：_____

姓名：_____成绩：_____

| 得 分 |
| --- |
| |

一、 单项选择题（每空 2 分，共 24 分）

1. 编译器报告“缺少运算符”错误是在__B__阶段，对常量进行类型转换是在__C__阶段，将可重定位机器码中相对地址修改为绝对地址是在__F__阶段。
   A．词法分析　　B．语法分析
   C．语义分析　　D．代码生成
   E．链接　　F．代码加载
2. 一个 C 语言数据库访问扩展系统的功能是将 C 程序中以##开头的 SQL 查询命令替换为对数据库查询 C 库函数的调用，则它是一种____A____。
   A．预处理器　　B．编译器
   C．汇编器　　D．链接器
3. 我们倾向于使用 DFA 而非 NFA 构造词法分析器，是因为____B____。
   A．DFA 空间占用优于 NFA
   B．DFA 时间复杂性优于 NFA
   C．以上皆对
   D．以上皆错
4. FIRST(α)={ε} 中的 ε 是 ___B____，正则表达式 ε 表示的是 _____D_____。
   A．单个符号
   B．长度为 0 的符号串
   C．空集
   D．包含一个符号串的集合
5. **aAA** 不是下面上下文无关文法的活前缀，原因是_____A_____。
   **S → aABe　　A → Abc | b　　B → d**
   A．不存在最右句型，其前缀是 aAA

B．aAA 是某个最右句型的前缀，但它在句柄左侧
C．aAA 是某个最右句型的前缀，但它的末尾超过了句柄末尾
D．以上皆错

6. 综合属性计算的依赖关系是父节点依赖孩子节点，所以综合属性的计算___B_____。
A．容易与预测分析法相结合
B．容易与算符优先分析算法相结合
C．以上皆对
D．以上皆错

7. 用预测分析法分析一个中缀表达式，当前栈顶是非终结符 **T**，输入缓冲区第一个终结符是+，则___D_____是**不恰当的**错误恢复方式。
A．将 **id** 压栈
B．将 **T** 弹出栈
C．将**( id )**插入到输入缓冲区开始
D．将+从输入缓冲区删除

8. 对 struct（record）类型的等价判定，下面说法**正确**的是___D_____。
A．C 语言采用结构等价判定，Pascal 语言采用名字等价判定
B．C 语言采用名字等价判定，Pascal 语言采用结构等价判定
C．C 语言和 Pascal 语言都采用结构等价判定
D．C 语言和 Pascal 语言都采用名字等价判定

9. 下面三地址码程序中，语句____B_____是公共子表达式。
**(1) x := y + z**
**(2) y := x – s**
**(3) z := y + z**
**(4) s := x – s**
A．(1)和(3)　　B．(2)和(4)
C．以上皆对　　D．以上皆错

| 得 分 |
| --- |
| |

二、 设计题（每题 6 分，共 24 分）**扣分标准：有一些瑕疵的话扣 1 分，每个部分有明显错误的话酌情扣分**

1. GB2312 编码将字符分为 94 个区（编号 1-94），汉字占据 16-87 区。每区 10 行 10 列，按行主次序编号 0-99，其中 0 和 95-99 是六个空位。每个字符采用两字节编码，高字节为区号，低字节为区内位号，区号和位号（先转换为十六进制）再分别加上 0xA0。例如，“王”在 45 区 8 行 5 列，因此其编码为 0xCDF5。设计正则表达式描述 GB2312 中汉字的编码。

**分数划分：区号部分 2 分，位号部分 4 分**

答：0x([B-E][0-F] | F[0-7]) (A[1-F] | [B-E][0-F] | F[0-E])

2. 英文单词不能以 I、U、V 或 J 结尾，例外是允许三个很古老的英语单词 I、YOU 和 THOU。设计正则表达式描述满足这一规则的字母串（大写）。
**分数划分：valid 2 分，主体 2 分，例外 2 分**
答：valid→[A-HK-TW-Z]
[A-Z]* valid | I | YOU | THOU

3. 设计非二义性上下文无关文法描述布尔表达式，其中基本布尔表达式为常量 **true**、**false** 和变量 **id**，布尔运算有 **and**、**or**、**not** 和**(**、**)**。
**分数划分：10 个候选式，错 1 个扣 0.5，满 5 个扣 3 分，错更多继续 1 个 0.5 累计**
答：E→E **or** T | T
T→T **and** F | F
F→**not** B | B
B→**(** E **)** | **true** | **false** | **id**

4. 设计上下文无关文法，生成正则表达式 **0(0 | 1)*0** 所描述的语言。
**分数划分：错 1 个候选式 1 分，全错 0 分。允许其他等价文法。**
S→0 A
A→0 A | 0 B | 1 A
B→**ε**

| 得 分 |
| --- |
| |

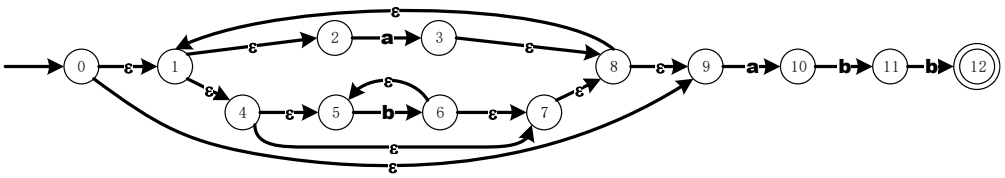
三、 （22 分）对下面的正则表达式。

**(a | b*)*abb**

1. 用 Thompson 构造法将其转换为 NFA。（7 分）

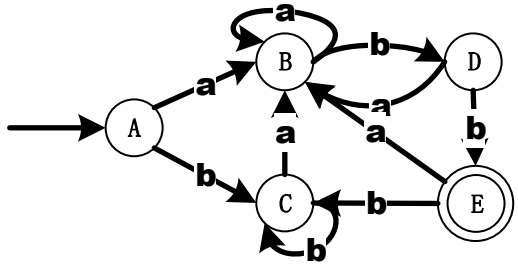
答：

**分数划分：基本正则表达式（a 和 b）共 1 分，连接、并集、闭包运算各 2 分。**




2. 用子集构造法将得到的 NFA 转换为 DFA，写出识别 ababbab 的状态转换序列和识别结果。（9 分）
**分数划分：子集构造法过程 6 分，状态转换图 1 分，状态识别和结果 2 分**
答：A=ε_clousure({0})={0,1,2,4,5,7,8,9}

ε_clousure(δ(A,a))={1,2,3,4,5,7,8,9,10}=B
ε_clousure(δ(A,b))={1,2,4,5,6,7,8,9}=C
ε_clousure(δ(B,a))=B
ε_clousure(δ(B,b))={1,2,4,5,6,7,8,9,11}=D
ε_clousure(δ(C,a))=B
ε_clousure(δ(C,b))=C
ε_clousure(δ(D,a))=B
ε_clousure(δ(D,b))={1,2,4,5,6,7,8,9,12}=E
ε_clousure(δ(E,a))=B
ε_clousure(δ(E,b))=C




A→B→D→B→D→E→B→D，结果是拒绝

3. 将 DFA 最小化（6 分）

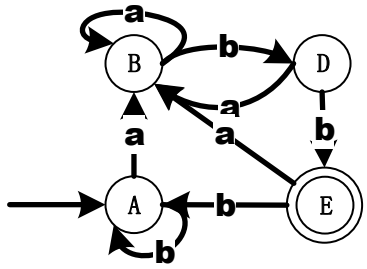
**分数划分：初始化分和三个步骤各 1 分，最终状态转换图 2 分**

答：初始划分{A, B, C, D} {E}

{A, B, C, D}→b {C, D, C, E}，因此划分为{A, B, C} {D}

{A, B, C}→b {C, D, C}，因此划分为{A, C} {B}

{A, C}不可再分，最终状态分组为{A, C} {B} {D} {E}




| 得 分 |
| --- |
| |

**四、** （14 分）对下面 if 语句的文法（**i**——if、**e**——else、**a**——代表其他语句的终结符，忽略了表达式）：

（1） 构造 SLR 分析表（7 分）

（2） 消解分析表中冲突，对 **i i a e i a e a** 进行语法分析（7 分）。

S → **i** S | **i** S **e** S | **a**

答：

(1)

**分数划分：12 个步骤 4 分，First、Follow 和 SLR 表 3 分**

closure({S'→.S})={ S'→.S, S→. **i** S, S→. **i** S **e** S, S→.**a** }=$I_0$

goto($I_0$,S)={ S'→S. }=$I_1$

goto($I_0$, **i**)={ S→**i** .S, S→**i** .S **e** S, S→. **i** S, S→. **i** S **e** S, S→.**a** }=$I_2$

goto($I_0$, **a**)={ S→**a** . }=$I_3$

goto($I_2$, S)={ S→**i** S. , S→**i** S. **e** S }=$I_4$

goto($I_2$, **i**)=$I_2$

goto($I_2$, **a**)=$I_3$

goto($I_4$, **e**)={ S→**i** S **e**. S, S→. **i** S, S→. **i** S **e** S, S→.**a** }=$I_5$

goto($I_5$, S)={ S→**i** S **e** S. }=$I_6$

goto($I_5$, **i**)=$I_2$

goto($I_5$, **a**)=$I_3$

First(S)={**i**, **e**, **a**} Follow(S)={**e**, **$**}

| | action | | | | goto |
|---|---|---|---|---|---|
| | **i** | **e** | **a** | **$** | S |
| 0 | s2 | | s3 | | 1 |
| 1 | | | | acc | |
| 2 | s2 | | s3 | | 4 |
| 3 | | r3 | | r3 | |
| 4 | | s5/r1 | | r1 | |
| 5 | s2 | | s3 | | 6 |
| 6 | | r2 | | r2 | |

(2)

**分数划分：消解冲突 2 分，分析过程 5 分**

对(4, e)表项的冲突，r1 表示 if stmt 归约，s5 表示 else 移进，将来 if stmt else stmt 归约，显然后者符合 else 与最近未匹配 if 相匹配的原则，因此保留 s5

| 栈 | 输入 | 输出 |
|---|---|---|
| $0 | **i i a e i a e a**$ | |
| $0 **i** 2 | **i a e i a e a**$ | |
| $0 **i** 2 **i** 2 | **a e i a e a**$ | |
| $0 **i** 2 **i** 2 **a** 3 | **e i a e a**$ | S→a |
| $0 **i** 2 **i** 2 S 4 | **e i a e a**$ | |
| $0 **i** 2 **i** 2 S 4 **e** 5 | **i a e a**$ | |
| $0 **i** 2 **i** 2 S 4 **e** 5 **i** 2 | **a e a**$ | |
| $0 **i** 2 **i** 2 S 4 **e** 5 **i** 2 **a** 3 | **e a**$ | S→a |
| $0 **i** 2 **i** 2 S 4 **e** 5 **i** 2 S 4 | **e a**$ | |
| $0 **i** 2 **i** 2 S 4 **e** 5 **i** 2 S 4 **e** 5 | **a**$ | |
| $0 **i** 2 **i** 2 S 4 **e** 5 **i** 2 S 4 **e** 5 **a** 3 | $ | S→a |
| $0 **i** 2 **i** 2 S 4 **e** 5 **i** 2 S 4 **e** 5 S 6 | $ | S→i S e S |

| | | |
|---|---|---|
| $0 **i** 2 **i** 2 S 4 **e** 5 S 6 | $ | S→i S e S |
| $0 **i** 2 S 4 | $ | S→i S |
| $0 S 1 | $ | acc |

| 得 分 |
|---|
| |

五、 （16 分）

（1）设计上下文无关文法描述类型表达式（注意，不是描述类型），基本类型为 **integer**、**char**、**real**，类型构造符为 **array**、**pointer**、**→**（函数）和 **X**（笛卡儿积）。（6 分）

（2）设计**语法制导定义**实现类型表达式转换为紧凑二进制编码表示，**integer**、**char**、**real** 编码分别为 **01**、**10**、**11**，**array**、**pointer**、**→**（函数）编码分别为 **01**、**10**、**11**。对类型表达式 **array(num, pointer(char)→integer)** （数组的索引集合用一个常数表示），画出其语法树，利用你设计的语法制导定义对语法树中节点计算属性值，给出最终的翻译结果（10 分）

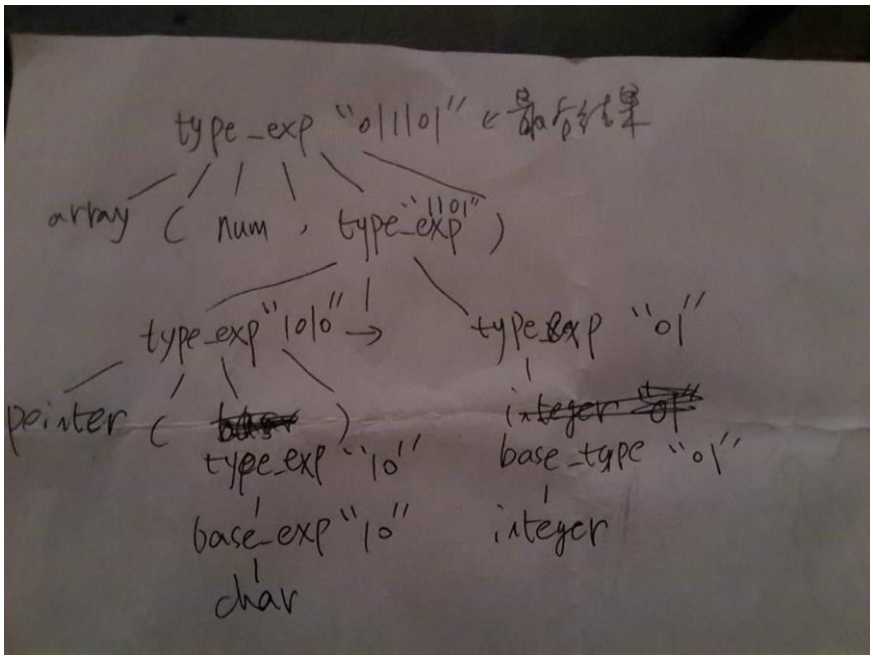
答：

（1）**分数划分：4 个类型构造符各 1 分，base_type 2 分**

**type_exp → base_type | array ( num , type_exp)**
**| pointer ( type_exp ) | type_exp 'X' type_exp**
**| type_exp '→' type_exp**
**base_type → integer | char | real**

（2）**分数划分：语法制导定义 6 分**

**type_exp → base_type { type_exp.code = base_type.code; }**
**| array ( num , type_exp$_1$) { type_exp.code = "01" || type_exp$_1$.code; }**
**| pointer ( type_exp$_1$ ) { type_exp.code = "10" || type_exp$_1$.code; }**
**| type_exp$_1$ 'X' type_exp$_2$ { type_exp.code = type_exp$_1$.code; }**
**| type_exp$_1$ '→' type_exp$_2$ { type_exp.code = "11" || type_exp$_2$.code; }**
**base_type → integer { base_type.code = "01"; }**
**| char { base_type.code = "10"; }**
**| real { base_type.code = "11"; }**


type_exp "0|1|0" 最终结果
array ( num , type_exp "1101" )
type_exp "1|0|0"
type_exp "0"
pointer ( base )
integer of
type_exp "10"
base_type "0"
base_exp "10"
integer
char