编译原理期中测验（2018）

学号：   姓名：   成绩：

一、**Mark each statement *true* or *false*   (2 points each， 6 cents)**

1. The same language token may be generated by many different regular expressions. （T）

2. To any regular expression, we can find a context-free grammar defining the same language. （T）

3. The LL(1) parsing algorithm parses an input string of tokens by tracing out the steps in a rightmost derivation. （F）

二、**Single Choice   (1 points each ， 5 cents)**

1、The concept (   ) is not related to the LL(1) parsing method.

    [A] Left-factoring   [B]. First set and follow set

    [C.] Left recursion removal   [D]. Shift and reduce

2、Which one below is not a part of a compiler?   （这个题目不是太好，就不扣分了）

    [A]   Symbol table   [B]   Assembler

    [C]   Code optimizer   [D]   Parser

3、In the production $A \to B\ \alpha\ C$, we have

  [A] Follow (C) $\subset$ Follow (A), First(B) $\subset$ First(A)

  [B] Follow (C) $\subset$ Follow (A), First(A) $\subset$ First(B)

  [C] Follow (A) $\subset$ Follow (C), First(B) $\subset$ First(A)

  [D] Follow (A) $\subset$ Follow (C), First(A) $\subset$ First(B)

4、IF one CFG grammar contains two non-terminals 'A','B' and two terminal 'a','b', where 'A' is the start symbol, then the Follow set of 'A' may be (   )

  [A] {a, b}   [B] {a, b, $}   [C] {a, b, ε}   [D] {a, b, B}

5、In the Top-Down Parsing, the action (   ) will never be used.

  [A] Shift   [B] Match   [C] Generate   [D] Accept

三、 question   (39 cents)

1、 Given the NFA for below for 0*(01)*0*, construct a   minimum state DFA: (8 cents)   （如果状态错的很多的话，就给个基本分**3**分）
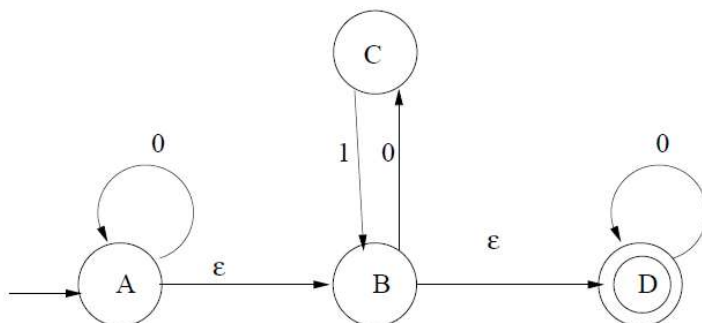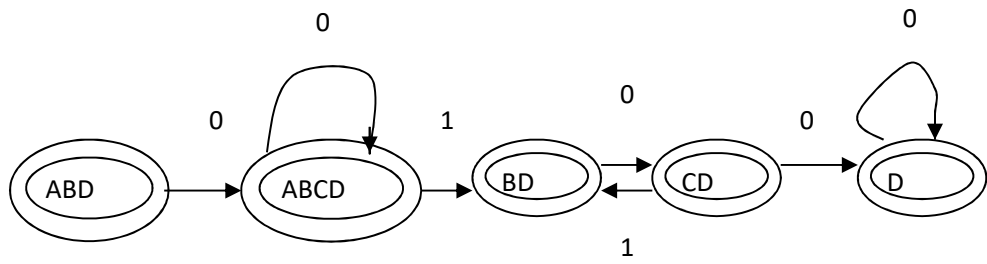
2、Given the follow grammar.　　（有的同学去掉了 **M** 再做下一步，应该也算对的 ）

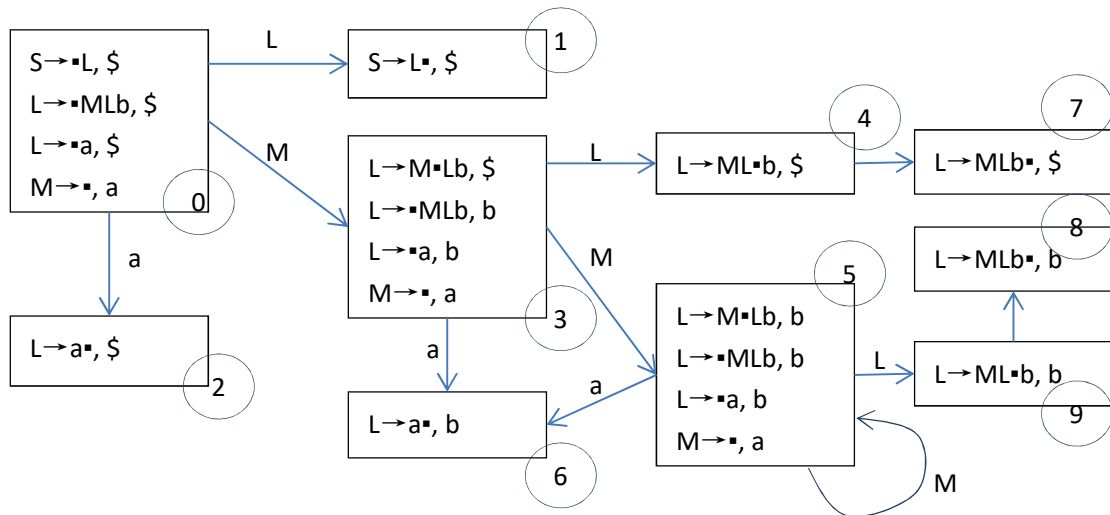   S→L

    L→MLb

    L→a

    M→ε

   (S is the start symbol.)

   Construct the LR(1) DFA for the grammar. **(10 cents) (**错一个状态扣 **1** 分**)**

Solution:

Solution:

  (1) The LR(1) DFA of this grammar is as follows:

3、(7　cents)

Give a RE and a CFG for:

$L = \{x \in \{0,1\}^* \mid x \text{ starts and ends with different symbols}\}$

$[a(a \mid b)^*b] \mid [b(a \mid b)^*a]$
$S \rightarrow aAb \mid bAa$
$A \rightarrow aA \mid bA \mid \varepsilon$

4、Consider the following grammar of simplified C declarations:

declaration → type   var-list

type → int | float

var-list → identifier, var-list | identifier

(a) Left factor this grammar.   （3 cents） （由于这个第一步没有做好影响下面的正确性，后面扣分可以适当少一点的）

(b) Construct First and Follow sets for the nonterminals of the resulting grammar.(6   cents) （错一个扣一分，扣完为止，注意：$没有的不扣分）

(c) Construct the LL(1) parsing table for the resulting grammar.   (5   cents)
（一个扣 **0.5** 分， 没有$这一列的不扣分，没有逗号的还是要扣分。）

d 部分没有放入题目中。

(a)   declaration → type var-list

      type → int | float

      var-list → identifier ID

      ID → , var-list | ε

(b)

FIRST(declaration) = { int, float }   FOLLOW(declaration) = { $ }

FIRST(type) = { int, float}   FOLLOW(type) = { identifier }

FIRST(var-list) = { identifier }   FOLLOW(var-list) = { $ }

FIRST(ID) = { , , ε}   FOLLOW(ID) = { $ }

(c)

	int	float	,	identifier	$
declaration	*declaration* → *type var-list*	*declaration* → *type var-list*			
type	*type* → int	*type* → float			
var-list				*var-list* → identifier *ID*	
ID			*ID* → , var-list		*ID* → ε

(d)

Parsing	Input	Action
$*declaration*	int x, y, z$	*declaration* → *type var-list*
$*var-list type*	int x, y, z$	*type* → int
$*var-list* int	int x, y, z$	match
$var-list	x, y, z$	*var-list* → identifier *ID*
$*ID* identifier	x, y, z$	match
$*ID*	, y, z$	*ID* → , var-list
$var-list ,	, y, z$	match
$var-list	y, z$	*var-list* → identifier *ID*
$*ID* identifier	y, z$	match
$*ID*	, z$	*ID* → , var-list
$var-list ,	, z$	match
$var-list	z$	*var-list* → identifier *ID*
$*ID* identifier	z$	match
$*ID*	$	ID → ε
$	$	accept