中山大学本科生期末考试

考试科目：《编译原理》（A 卷答案）

学年学期：2016 学年第一学期　　　　　　　　姓　　名：______________

学　院/系：数据科学与计算机学院/软件工程学　　号：______________

考试方式：闭卷　　　　　　　　　　　　　　　年级专业：______________

考试时长：120 分钟　　　　　　　　　　　　　班　　别：______________

警示《中山大学授予学士学位工作细则》第八条："考试作弊者，不授予学士学位。"

------------以下为试题区域，共九道大题，总分 100 分，考生请在答题纸上作答------------

1. Write regular expressions for the following languages (8 points).

a) **All strings over {0, 1} that do not contain consecutive zeros (4 points).**

(0?1)*0?　　(01 | 1)*(0 | ε).

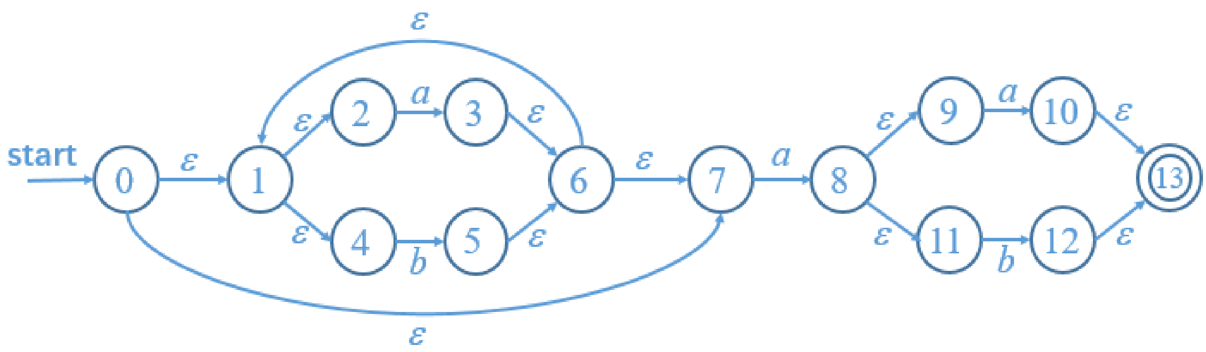
b) **All strings representing a nonnegative binary number (without leading zeros) which is a multiple of 8 (4 points).**

0 | 1(1|0)*000

2. Consider the following regular expression (17 points):

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

a) **Based on the Thompson Algorithm, construct the NFA from the above regular expression (8 points).**



b) Convert the NFA into a DFA with minimum number of states (9 points).

ε-closure({0}) = {0, 1, 2, 4, 7}

ε-closure(move({0, 1, 2, 4, 7}, ‘a’)) = ε-closure({3, 8}) = {1, 2, 3, 4, 6, 7, 8, 9, 11}

ε-closure(move({0, 1, 2, 4, 7}, ‘b’)) = ε-closure({5}) = {1, 2, 4, 5, 6, 7}

ε-closure(move({1, 2, 3, 4, 6, 7, 8, 9, 11}, ‘a’)) = ε-closure({3, 8, 10}) = {1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 13}

ε-closure(move({1, 2, 3, 4, 6, 7, 8, 9, 11}, ‘b’)) = ε-closure({5, 12}) = {1, 2, 4, 5, 6, 7, 12, 13}

ε-closure(move({1, 2, 4, 5, 6, 7}, ‘a’)) = ε-closure({3, 8}) = {1, 2, 3, 4, 6, 7, 8, 9, 11}

ε-closure(move({1, 2, 4, 5, 6, 7}, ‘b’)) = ε-closure({5}) = {1, 2, 4, 5, 6, 7}

ε-closure(move({1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 13}, ‘a’)) = ε-closure({3, 8, 10}) = {1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 13}

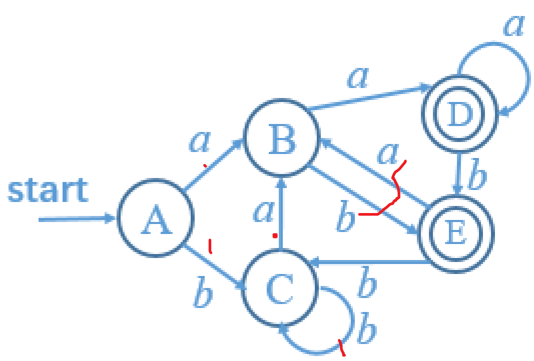
ε-closure(move({1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 13}, ‘b’)) = ε-closure({5, 12}) = {1, 2, 4, 5, 6, 7, 12, 13}

ε-closure(move({1, 2, 4, 5, 6, 7, 12, 13}, ‘a’)) = ε-closure({3, 8}) = {1, 2, 3, 4, 6, 7, 8, 9, 11}

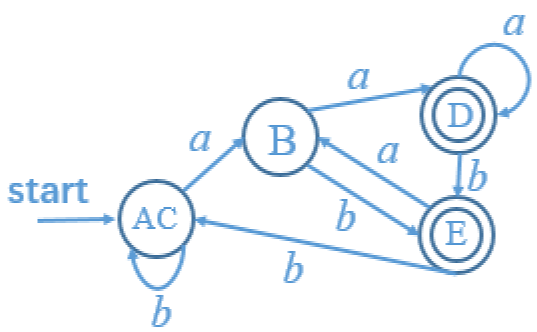
ε-closure(move({1, 2, 4, 5, 6, 7, 12, 13}, ‘b’)) = ε-closure({5}) = {1, 2, 4, 5, 6, 7}

NFA states	DFA state	a	B
{0, 1, 2, 4, 7}	A	B	C

{1, 2, 3, 4, 6, 7, 8, 9, 11}	B	D	E
{1, 2, 4, 5, 6, 7}	C	B	C
{1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 13}	D	D	E
{1, 2, 4, 5, 6, 7, 12, 13}	E	B	C



Minimized DFA:



3. Consider the following CFG (9 points):

S → +SS | -SS | a

and the string +a-aa.

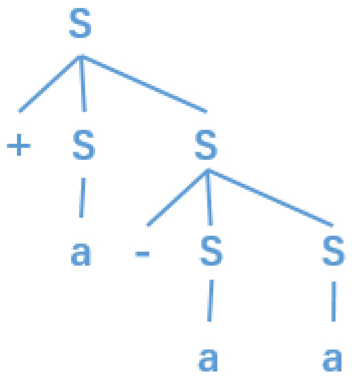
a) Give a leftmost derivation for the string (3 points).

S => +SS => +aS => +a-SS => +a-aS => +a-aa

b) **Give a rightmost derivation for the string (3 points).**

S => +SS => +S-SS => +S-Sa => +S-aa => +a-aa

c) **Give a parse tree for the string (3 points).**



4. **Write CFGs for the following languages (8 points):**

a) **L={$a^n b^m c^n$ | $m \geq 0$, $n \geq 1$} (4 points).**

S → aSc | aAc

A → bA | ε

b) **All strings over {a, b} that begin and end with the same letter (4 points).**

S → aAa | bAb | a | b

A → aA | bA | ε

5. **Consider the following CFGs (18 points):**

S → i(E)t(E)X | E

X → e(E) | ε

E → aY | bY

Y → c(E) | ε

where i, t, e, a, b, c, (and) are terminals.

a) Compute the FIRST and FOLLOW sets for all non-terminals (8 points).

Non-terminal	FIRST	FOLLOW
S	i, a, b	$
X	e, ε	$
E	a, b	), $
Y	c, ε	), $

b) Construct an LL(1) parsing table for the grammar (8 points).

	i	t	e	a	b	c	(	)	$
S	S→i(E)t(E)X			S→E	S→E				
X			X→e(E)						X→ε
E				E→aY	E→bY				
Y						Y→c(E)		Y→ε	Y→ε

c) Is this grammar LL(1)? Why? (2 points)

Yes, because no conflicts can be found in the LL(1) parsing table.

6. Consider the following CFG (12 points):

S→aAD | aBe | bBS | bAe

A→g

B→g

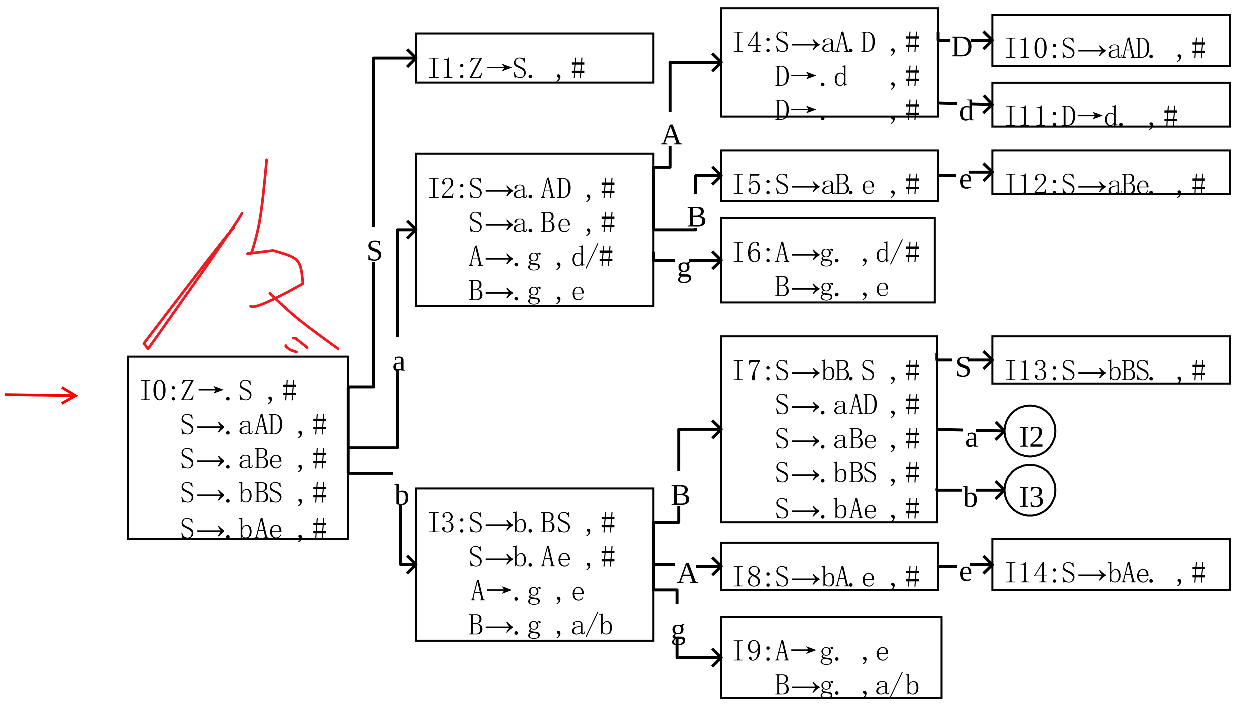
D→d | ε

augment the grammar and construct the LR(1) sets of items for the augmented grammar.

首先拓广 G[S]为 G[Z]:

0: Z → S	1: S → aAD	2: S → aBe
3: S → bBS	4: S → bAe	5: A → g
6: B → g	7: D → d	8: D → ε

构造G[Z]的LR（1）项目族为:



7. The following grammar generates binary strings and their complements (10 points).

F → B

| ¬B

B → B0
| B1
| 0
| 1

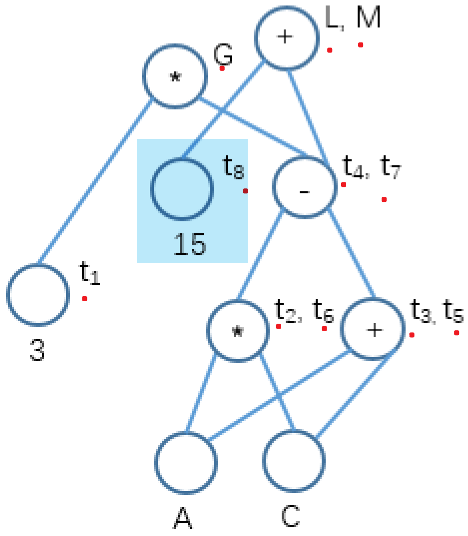
The value of a (non-negated) string is just the decimal value of the binary number the string represents; the value of a negated string is the decimal value of the string with 1's replaced by 0's and 0's replaced by 1's. For example, the value of 010 is 2 and ¬010 is 5. Design a syntax-directed definition (SDD) for the above grammar such that the non-terminal F has an attribute F.val which keeps the value of an input string generated by F. Please do NOT modify the grammar.

产生式	语义规则
F → B	B.c = false, F.val = B.val
F → ¬B	B.c = true, F.val = B.val
B → $B_1$0	B_1.c = B.c, B.val = B_1.val * 2 + (B.c ? 1 : 0)
B → $B_1$1	B_1.c = B.c, B.val = B_1.val * 2 + (B.c ? 0 : 1)
B → 0	B_1.c = B.c, B.val = (B.c ? 1 : 0)
B → 1	B_1.c = B.c, B.val = (B.c ? 0 : 1)

8. Consider the following basic block (10 points):

1)	$t_1 = 3$
2)	$t_2 = A * C$
3)	$t_3 = A + C$
4)	$t_4 = t_2 - t_3$
5)	$G = t_1 * t_4$
6)	$t_5 = A + C$
7)	$t_6 = A * C$
8)	$t_7 = t_6 - t_5$
9)	$t_8 = t_1 * 5$
10)	$L = t_8 + t_7$
11)	$M = L$

a) Construct the DAG of the above basic block (5 points).



b) Assume that only G, L and M will be used after the basic block. Give the optimized three-address statement sequence (5 points).

1)	t_2 = A * C
2)	t_3 = A + C
3)	t_4 = t_2 – t_3
4)	G = 3 * t_4
5)	L = 15 + t_4
6)	M = L

9. Consider the following fragment of three-address instructions (8 points):

(1)		b := 1
(2)		b := 2
(3)		if w <= x goto B
(4)		e := b
(5)		goto B
(6)	A:	goto D
(7)	B:	c := 3
(8)		b := 4
(9)		c := 6
(10)	D:	if y <= z goto E
(11)		goto End
(12)	E:	g := g + 1
(13)		h: = 8
(14)		goto A
~~(15)~~	~~End:~~	~~h := 9~~

Please partition these three-address instructions into basic blocks, and draw the control flow graph. You may draw the resulting graph directly, but you must mark

each node by number n~m indicating that the corresponding basic block consists of instructions n through m, inclusive.