94
106

BIRZEIT UNIVERSITY
College of Information Technology
Computer Science Department

Name : Ali Samara Number : 1090604
COMP 439 Midterm Exam 20-11-2012

[1] Describe the language generated or accepted by the following CFG and FSAs:

(a) $S \rightarrow 0S1 \mid \lambda$

$L(G) = \{ 0^n 1^n \mid n \geq 0 \}$

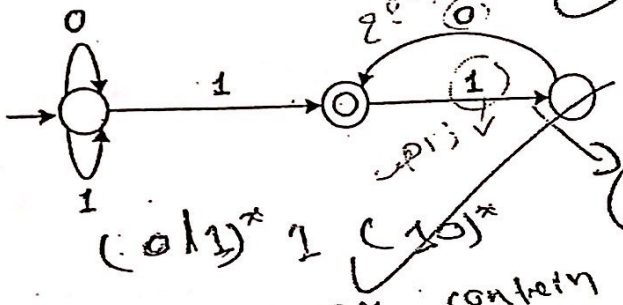
any string of a's and b's start and end with a or b

(b) $S \rightarrow 0S1 \mid 1S0 \mid \lambda$

$L(G) = \{$ 0000 1111, 1111 0000, 10101010 $\}$

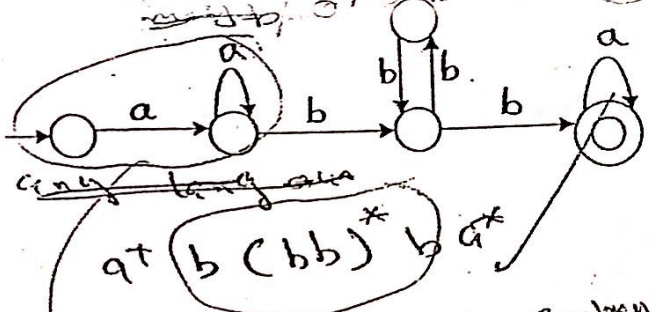
any string contains 0's or 1's in the same number and if it start with 0 it must end with 1 or if it start with 1 it must end with 0

(c)



$(0|1)^* 1 (10)^*$

(d)



any string contain b's or a's it starting with a or b and it may end by b or a and it has any string of a's and b's

$a^+ b (bb)^* b a^*$

$a^+ (bb)^+ a^*$

$b(b^*) = b^+$

$b(bb)^* b = (bb)^+$

any string start with a and it may contain's b's and a's must end with b or a

$a(a)^* (b(bb)^* b) a^*$

$= a^+ (bb)^+ a^*$

Scanned with CamScanner

[2] Given the grammar,

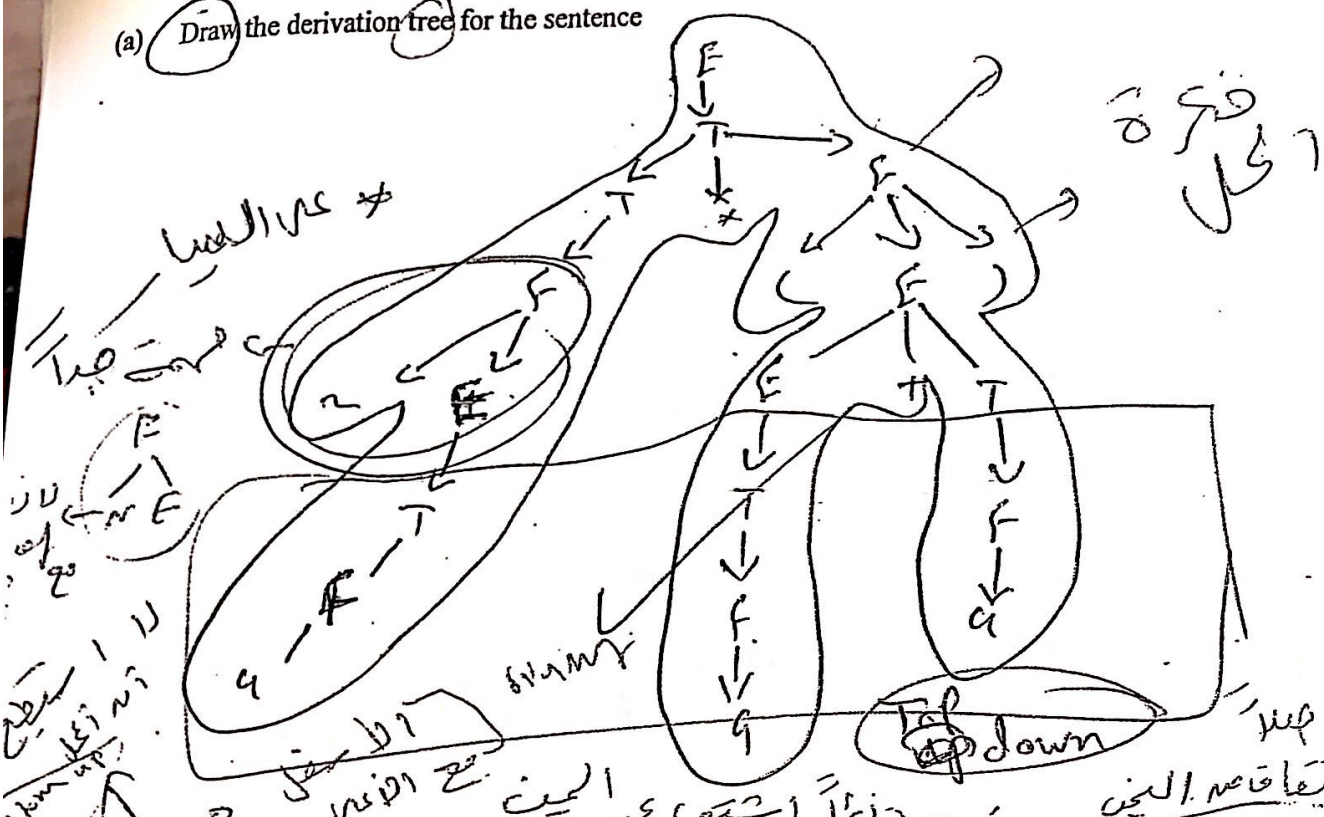
$E \rightarrow E+T \mid T$

$T \rightarrow T*F \mid F$

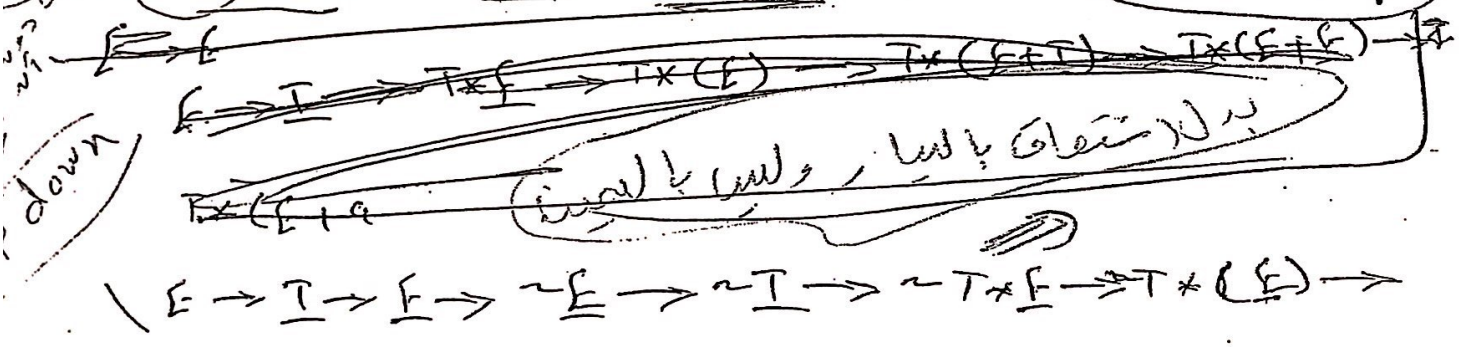
$F \rightarrow (E) \mid a \mid \sim E$

Given the sentence ~a*(a+a)

(a) Draw the derivation tree for the sentence



(b) In bottom-up parsing, show the **handle** in each reduction step.



$F \rightarrow T \rightarrow F \rightarrow \sim E \rightarrow \sim T \rightarrow \sim T*F \rightarrow T*(E) \rightarrow$

$\sim T*(E+T) \rightarrow \sim T*(E+F) \rightarrow \sim T*(E+a) \rightarrow T*(T+a) \rightarrow$

$\sim T*(F+a) \rightarrow \sim T*(a+a) \rightarrow \sim F*(a+a) \rightarrow a*(a+a)$

the handle

$F \rightarrow T \rightarrow F \rightarrow \sim E \rightarrow \sim T \rightarrow \sim T*F \rightarrow \sim T*(E) \rightarrow \sim T*(E+T)$

$\sim T*(E+F) \rightarrow \sim T*(E+a) \rightarrow \sim T*(T+a) \rightarrow \sim T*(F+a) \rightarrow$

$\sim T*(a+a) \rightarrow \sim F*(a+a) \rightarrow \sim a*(a+a)$

5] (a) Given a sample program(sentence) in some programming language L(G) where :

V_T = { var , int , exit , exp , ; , = , [,] }

```
var
   int name;
   int name;
   . . . .
[  name = exp;
   name = exp;
   }
   name = exp;
] exit
```

Write down the production rules P , that generates this language.

(b) Given the CFG :

S → A1B

A → 0A | λ

B → 1B | λ

(i) Define L(G) ?

$L(G) = \{ 0^m 1^n \mid m \geq 0 , n \geq 1 \}$

(ii) Show a leftmost and a rightmost derivation for the sentence 00011

[4] Given the following DFSA.

- Reduce it to FSA with **minimum** states using tabular methods.
- Draw the final FSA.
- The underline states are final states.

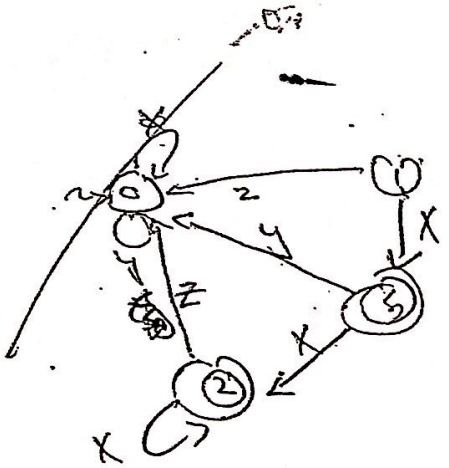
δ	x	y	z
0	3	3	
1	5	-	4
2	6		4
3	5	4	
4	0	4	
5	6	3	
6	2		4

	x	y	z
(2,6)	(2,6)		(4,4)
(0,3)	(3,3)	(3,4)	
(0,4)	(3,0)	(3,4)	
(3,4)	(3,0)	(3,4)	

so 2≡6 , 0≡3≡4.

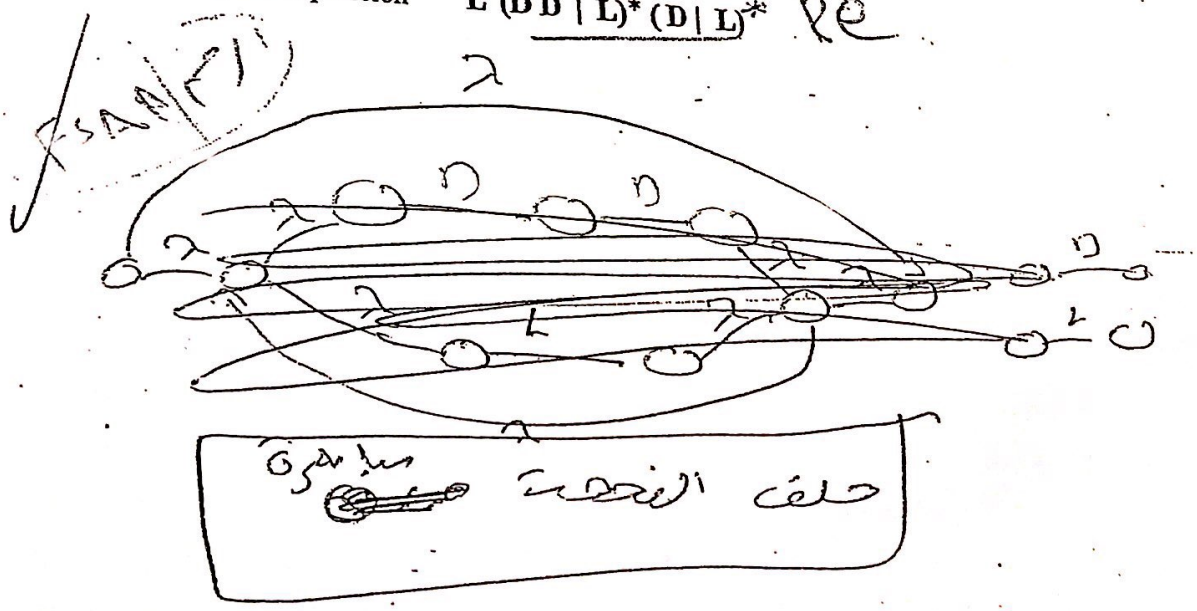
-new state

	x	y	z
0	0	0	
1	5		0
2	2		6
5	2	0	



but ther state un accessable

(b) Draw a FSA that accepts regular expression **L (D D | L)* (D | L)*** RE



حلف الجملة صحيحة

[5] (a) Given the grammar G with productions :

$A \rightarrow \alpha$
$A \rightarrow \beta$
$A \rightarrow \lambda$

State explicitly the conditions so that the above grammar is LL(1).

(b) Given the grammar G,

$S' \rightarrow S\,\$$
$S \rightarrow (L) \mid a$
$L \rightarrow SA$
$A \rightarrow ,SA \mid \lambda$

Using the formal definition only, Examine if G is LL(1) ?
Do you think it is LL(2)? . Use the formal definition only.

Consider the grammar:

(1) S' → S$
(2) S → ABA
(3) A → Bc
(4) A → dA
(5) A → λ
(6) B → eA

(a) Fill in the table below with FIRST & FOLLOW sets :

	FIRST	FOLLOW
S'	e, d	—
S	e, d	$
A	e, d, λ	c, e, d, $
B	e	c, e, d, $

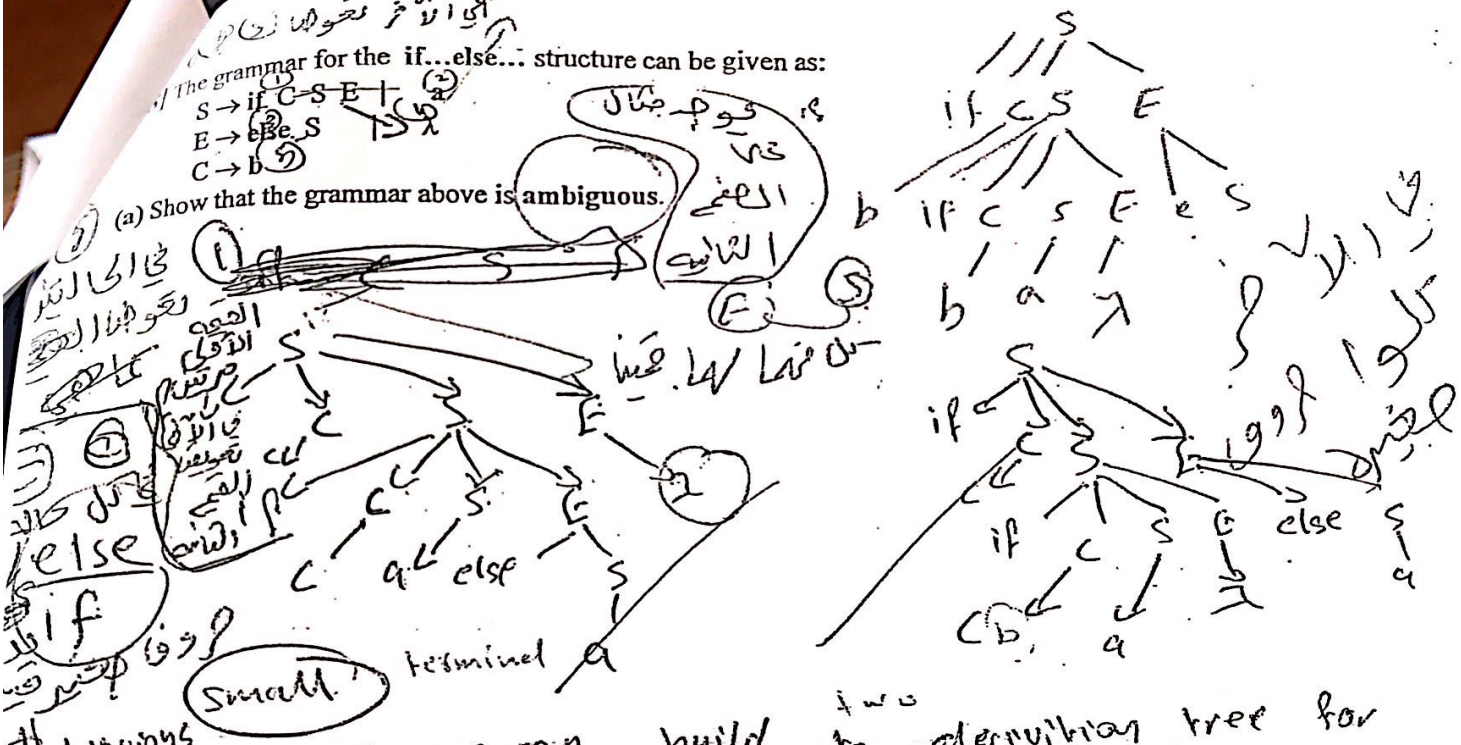
(b) Fill in the LL(1) parsing table

	c	d	e	$
S'		1	1	1
S		2	2	
A	5	4 5	3 5	5
B			6	

The grammar for the if...else... structure can be given as:

$S \rightarrow$ if C S E | a
$E \rightarrow$ else S
$C \rightarrow$ b

(a) Show that the grammar above is ambiguous.



ambiguous because we can build two derivation tree for the same entrance

(b) Build the LL(1) parsing table. Is the grammar LL(1) ? why? How can you fix the problem without changing the grammar structure? Explain.

	if	b	a	else
S	1		1	
E				2, 3
C		4		

the grammer is not LL(1) because there conflict

we can avoid this problem by constrain the else for nearest if

multiple production

Grammer

multiple production

BIRZEIT UNIVERSITY
College of Information Technology
Computer Science Department

1102040

Name : Jamal Abed Alnaser Number : 1102040

COMP 439 AL Midterm Exam 11-04-2013

[1] Reduce the following NDFSA to DFSA with fewest states using tabular methods:

	a	b	λ
1	2,4	3	2,3
2	4		2
3	2	3	
④	2,4	3,4	1,2,3
(2,4)	2,4	3,4	
(3,4)	2,4	3,4	

δ	a	b	λ
1	2		2,3
2	4		2
3	2	3	
④		4	1

final

find

(2,4) ≡ 5
(3,4) ≡ 6 → Node

	a	b
1	4,5	3
2	4	
3	2	3
④	4,5	6,4
5	5	6
6	5	6

(1,2,3)
(1,2) (1,3)
(2,3)

(4,5,6)
(4,5) (4,6)
(5,6)

4 ≡ 5 ≡ 6

reduce

	a	b
(1,3)	(5,5)	(3,3)
(4,5)	(5,5)	(6,6)
(4,6)	(5,5)	(6,6)
(5,6)	(5,5)	(6,6)

تحذف 5,6
وتكون بدل
(a ← 5,6

(a) In some programming language, the structure of the CASE statement as follows:

```
case variable-name of
   int-value : statement ;
   int-value : statement ;
       .
       .
   int-value : statement ;
   else: statement ;
end
```

Write down the production rules that generate the CASE statement above.(words in **bold** are reserved words)

program → declaration "case of" "else" "end"

declaration → declitem ; declarations | ε

declitem → type name-list

type → "int" | "float" | "UDN"

name-list → ~~name-names~~ | ε

morenames → name-list | ε

case of → ~~statement~~ ; statement ; statement list

statement → "type" - value : statement

else

"end"

(b) Do you think the following grammar is ambiguous?

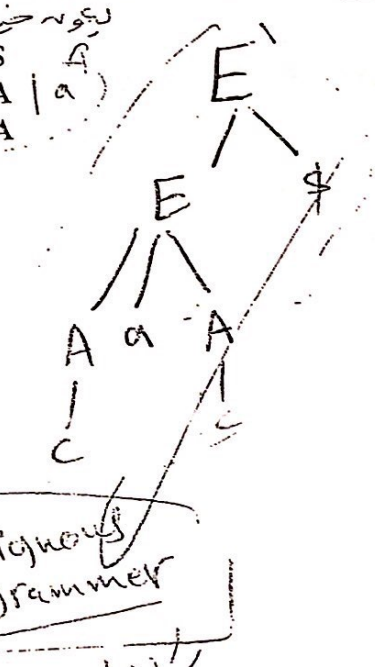
ambiguous يكون

$E' \rightarrow E\,\$$

$E \rightarrow AaA \mid a$

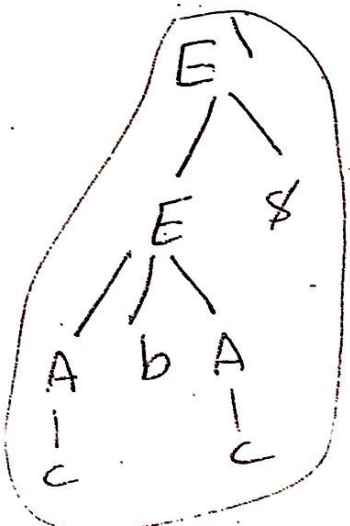
$E \rightarrow AbA$

$A \rightarrow c$



acb$

not ambigious grammer



cbc$

Given the grammar G below:

$A' \to AS$

$A \to xA \mid yA \mid y$

(a) Using the **formal definition only**, Examine if G is LL(1)

$FIRST(A) = [x, y]$

$FIRST(A') = (x, y)$

$Follow(A) = [\$]$

$FIRST(xA) \cap FIRST(yA)$

$= [x] \cap [y] = \emptyset$

i) $FIRST(xA) \cap FIRST(y)$

$[x] \cap [y] = \emptyset$

$FIRST(yA) \cap FIRST(y)$

$= \{y\} \neq \emptyset$

is not LL(1) grammer.

(b) Using the **formal definition only**, Examine if G is LL(2)

$FIRST_2(xA) \cap FIRST_2(yA)$

$= [xx, xy] \cap [yx, yy] = \emptyset$

$FIRST_2(xA) \cap FIRST_2(y)$ (y $\$$ y)

$= [xx, xy] \cap [y] = \emptyset$ ✓

$FIRST_2(yA) \cap FIRST_2(y)$

$= [yx, yy] \cap [y\$]$

$= \emptyset$

G is LL(2) grammer

(c) If we replace the production $A \to y$ by $A \to \lambda$, How this will affect (a) and (b) ?

$A' \to AS$

$A \to xA \mid yA \mid \lambda$

$FIRST(A) = [x, y, \lambda]$

$FIRST(A') = [x, y, \$]$

$follow(A) = [\$]$

$follow_2(A) = [\$\$]$

LL(1) $\Rightarrow$ LL(2)

1) $FIRST(xA) \cap FIRST(yA)$

$[x] \cap [y] = \emptyset$

2) $FIRST(xA) \cap follow(A)$

$[x] \cap [\$] = \emptyset$

3) $FIRST(yA) \cap follow(A)$

$[y] \cap [\$] = \emptyset$

G is LL(1) grammer

4) $FIRST_2(xA) \cap FIRST_2(yA)$

$= [xx, xy] \cap [yx, yy] = \emptyset$

5) $FIRST_2(xA) \cap Follow_2(A)$

$[xx, xy] \cap [x\$] = \emptyset$

6) $FIRST_2(yA) \cap Follow_2(A)$

$= [yx, yy] \cap [\$\$] = \emptyset$

G is LL(2) grammer

Consider the grammar:

(1) S' → S$
(2) S → [SX]
(3) S → a
(4) X → +SY
(5) X → Yb
(6) X → λ
(7) Y → -SXc
(8) Y → λ

(a) Fill in the table below with FIRST & FOLLOW sets :

	FIRST	FOLLOW
S'	[, a	$
S	[, a	$, +, -, b,]
X	+, -, b, λ	], c
Y	-, λ	b,], c

(b) Fill in the LL(1) parsing table

	a	b	c	+	-	[	]	$
S'	1					1		
S	3					2		
X		6,5		4	5		6,5	
Y		8	8		7		8	

(c) Is the above grammar LL(1) ? why ? If so, Show how to parse the sentence [a+a]

1) FIRST([SX]) ∩ FIRST(a)
= {[} ∩ {a} = ∅
2) FIRST(+SY) ∩ FIRST(Yb)
= {+} ∩ {-, λ} = ∅
3) FIRST(+SY) ∩ follow(X)

not is LL(1) grammer
Because exist multiple entries.

DSI

S' → S$ → [SX]$ →
[S+SY]$ → [S+S]$ →
[S+a]$ → [a+a]$

(a) Given the languages $L = \{aa, \lambda\}$, $M = \{b, cc\}$ defined over $V_T = \{a, b, c\}$

(i) Compute $L^2 M$

$L^2 M = [(aaaa, aa, aa, \lambda) \cdot (b, cc)]$

$= [aaaab, aaab, aaab, b, aaaacc, aacc, aacc, cc]$

$L^2 = [aaaa, aa, aa, \lambda]$

$(aa, \lambda)(aa, \lambda)$

(ii) Compute $L^2 \varphi$

$L^2 \varphi = \phi$

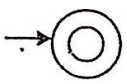
(iii) Compute $L^2 \{\lambda\}$

$= [aaaa, aa, aa, \lambda] \cdot [\lambda]$

$= [aaaa, aa, aa, \lambda] = L^2$

(b) What is the language recognized by the following FSA:

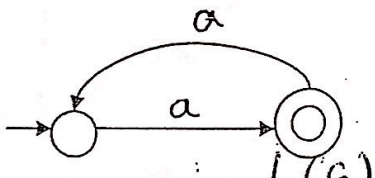
(i)



$L(G) = [a^-]$

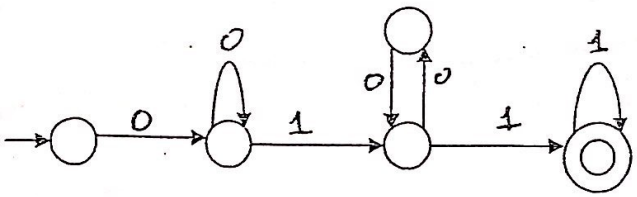
$L(G) = [\lambda]$

(ii)



$L(G) = (aa)^*$

(iii)

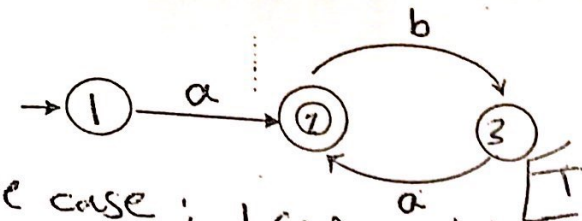


$G = (0)^+ | (00)^* (1)^+$

(a) Using Mathematical Induction, prove that the lang. accepted by the machines:

$L(M) = \{ a(ba)^n , n \geq 0 \}$

	a	b
1	2	
(2)		3
3	2	



The property is true)

1) base case; $L(M)$ is true when $n=0$

$L(M) = a(ba)^0 = a(ba)$

2) Inductive step: assume $L(M) = a(ba)^0$

then prove that

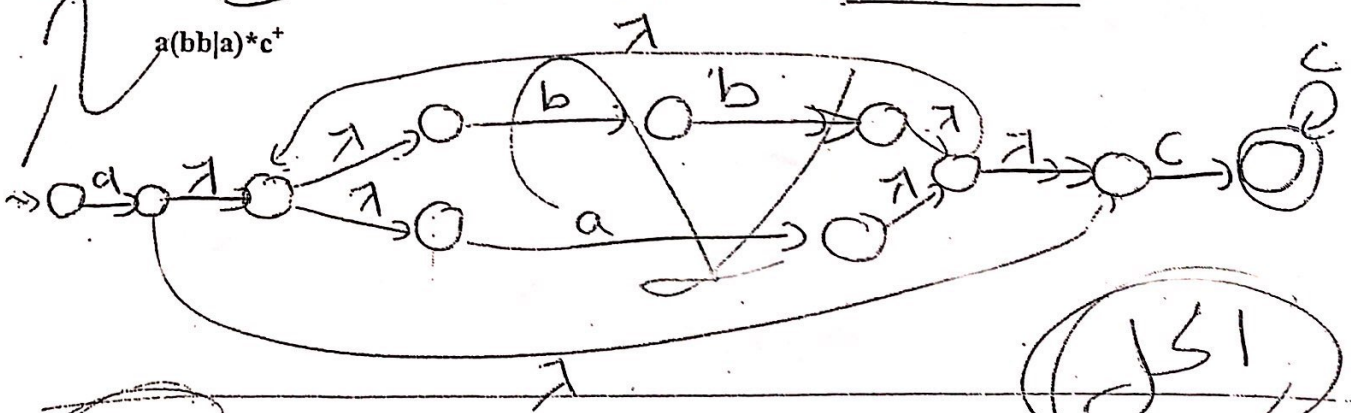
$L(M+1) = a(ba)^{A+1} = a(ba)^n \cdot a(ba)^1$

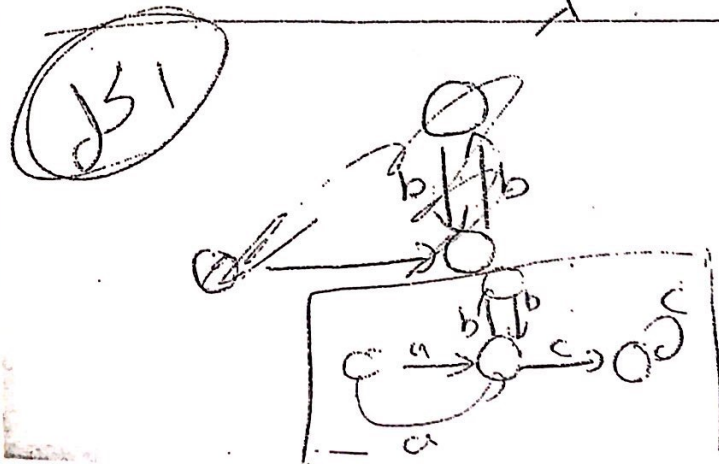
$a(ab)^*$

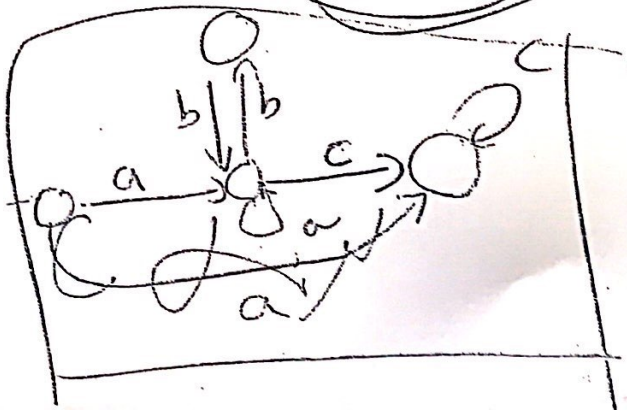
$L(m+1) = L(M)$

(b) Draw λ-free FSA that recognizes (accepts) the following Regular Expressions :

$a(bb|a)^*c^+$







BIRZEIT UNIVERSITY

College of Engineering & Technology
Computer Science Department
Midterm Exam

Comp439 Spring 17/18

[1] (a) Explain briefly the meaning of **Machine Readability** of a PL.

The ability to develop and write an Algorithm to translate the PL in an unambigious and finite way.

(b) Explain Briefly the meaning and give examples of **multiplicity** in a PL.

The ability to use PL in many plat forms like Java

(c) What is the major **one** difference between **compiler** and **interpreter.**

Compiler generate object code while interpreter generate Intermidiate code.

(d) Two advantages of a **compiler** are :

1-. Generate object code

2- Faster

while two advantages of an interpreter are :

1- Generate portable Intermidiate code.

2- easy to implement.

chapter #3

(a) Given the following function in CLISP

```
(defun func ( n )
  ( if ( = n 1) 1
     ( if ( = n 2) 1
        (+ ( func ( - n 1)) ( func ( - n 2 ))))))
```

What is the output of the function call (func 8)? Justify your answer by **tracing** the function call.

> (func 8)

Func(8) = 21

Func(7) + Func(6)

Func(6) + Func(5)

Func(5) + Func(4)

Func(5) + Func(4)

Func(4) + Fun(3)

Func(4) + Func(3)

Func(3) + Func(2)

Func(2) + Func(1)

$(n-2)!$

(b) In C , given the following function :

```
int F( int m , int n)     // m , n > 0
{  while ( m != n)
       if ( m > n)    m = m - n;
       else           n = n - m;
   return m; }
```

Rewrite this function in CLISP. What do you think the function do? Gcd

Greatest common devison

```
(defun Gcd ( m n)
 ( if (= m n) m
  ( if ( < m n)
   (+ ( Gcd ( m - n) n)
    (+ Gcd ( n (n - m)))))))
```

/6

Given the following simple grammar in BNF:

exp ⟶ (list)' | a

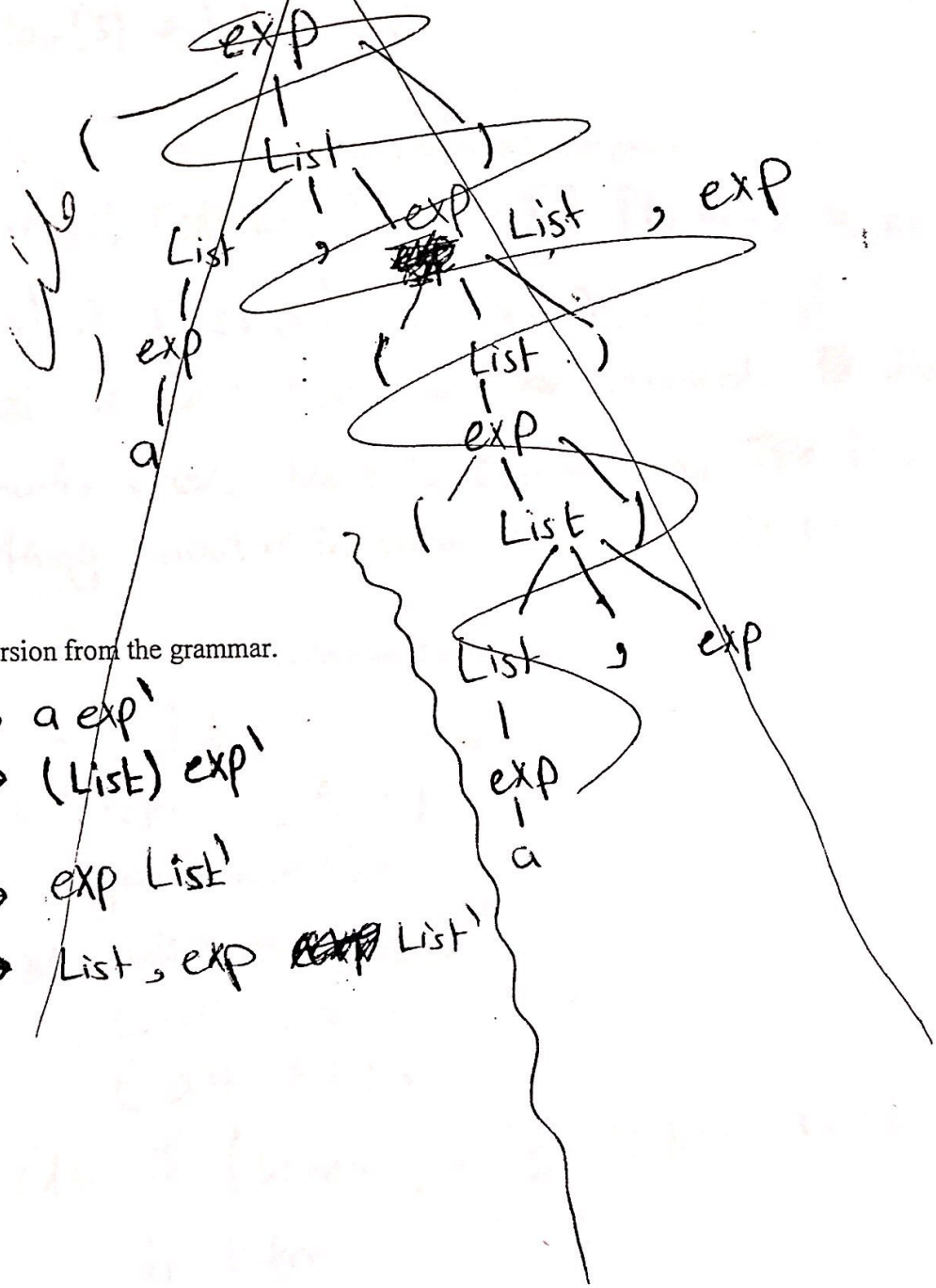
list ⟶ list , exp | exp

Where : $V_N = \{exp , list\}$, $V_T = \{(,), , , a\}$

a) Rewrite these productions in EBNF.

List → exp (, exp)*

(b) Draw the derivation tree for the sentence (a, ((a, a) ,a))



(c) Eliminate left recursion from the grammar.

exp → a exp'

exp' → (List) exp'

List → exp List'

List' → List , exp List'

Given the grammar :

$G \rightarrow S\$$
$S \rightarrow aSA \mid \lambda$
$A \rightarrow bS \mid c$

$V_N = \{G, S, A\}$, $V_T = \{a, b, c, \$\}$

(a) Compute:

FIRST(A) = $\{b, c\}$

FIRST(S) = $\{a, \lambda\}$

FIRST(G) = $\{a, \lambda\}$

FOLLOW(S) = $\{\$, b, c\}$

FOLLOW(A) = Follow(S) = $\{\$, b, c\}$

(b) Do you think this grammar is suitable (ملائم) for a recursive descent parsing ? Explain your answer.

$First(aSA) \cap Follow(S) = \{a\} \cap \{\$, b, c\} = \emptyset$

$First(bS) \cap First(c) = \{b\} \cap \{c\} = \emptyset$

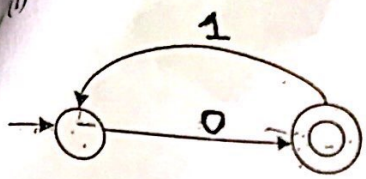
As we see we checked the statements where there is choices and there is nothing similar in them so it's suitable.

(c) If so, write a recursive descent parser for the non-terminal S only.

```
Func S (){
    if (token == "a")
        getToken();
    Call S();
    Call A();
else if (token != "$" || token != "b"
    || token != "c")
    Error();
```

(a) What is the language recognized by the following FSA:

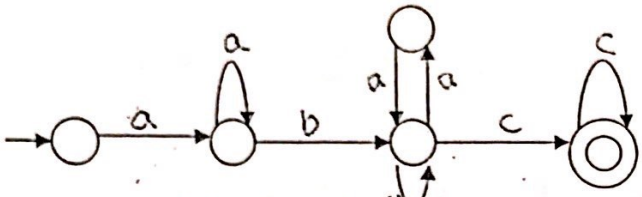
(i)



$(0(1)^*)$

$0(10)^*$ —3

(ii) $0(10)^*$



$a^+b(aa|b)^*c^{1+}$

$a^*b(aa|b)^*c^*$ —1

(b) The grammar for the **if...else...** structure can be given as:

$S' \rightarrow S\$$ ①
$S \rightarrow iCSE$ ② | a ③
$E \rightarrow eS$ ④ | λ ⑤
$C \rightarrow c$ ⑥

$V_N = \{S', S, C, E\}$, $V_T = \{i, a, c, e, \$\}$

Build the LL(1) parsing table. Is the grammar LL(1) ? why ? Fix it without changing the grammar.

V_N \ V_T	i	a	c	e	$
S'	1	1			
S	2	3			
E				4,5	5
C			6		

~~We will add a delimeter (d) at the end of the statement (s). there will be no conflict if we build the table again.~~

There are a conflict in the table so It's not LL(1).

We will add a delimeter (d) at the end of the statement(s) there will be no conflict if we build the table again.

(a) Given the grammar:

S → C ;
C → A : T
A → N | N , A
N → n
T → t

Given the sentence **n,n:t;** What are the **Handles** in a bottom-up parsing ?

S → C; → A : T; → A : t; → N , A : t;

→ N , N : t; → N , n : t; → n , n : t;

[ ] Are the handles ✓

(b) What is the problem with the following **right associative** grammar? Explain in full.

E → T + E | T
T → F * T | F
F → (E) | a

The problem here is in the precedence in the real world we make the mathematic operation from left to the write whit Focous of precedence. but this differ from our imagination.