Each Question 12 points
Exam out of 72 points

KEY

BIRZEIT UNIVERSITY
College of Engineering & Technology
Computer Science Department

(12)

Comp439 | Midterm Exam | 22/11/ 2018

4 [1] (a) What are the programming languages views? Explain.

1- Designer (Inventor of the PL)

2- Implementor (who develops & writes the compiler)

3- User (who writes programs in this PL)

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

The PL should behave same thing in similar contexts.

EX: In IBM machines, Addition is performed:

A register, memory
AR register1, register2 > Less orthogonal

But in VAX machines, there is only one instruction.

ADDL operand1, operand2 > more orthogonal better

4 (c) What are the programming languages paradigms. Give example on each.

1- Imperative (procedural) paradigm: Pascal, C

2- Functional Paradigm: Lisp

3- Logical Paradigm: Prolog

4- Object Oriented Paradigm: Java

12

[2] (a) Write a function in Clisp, **min**(x y) which computes the minimum value of x and y.

```
> (defun min (x y)
    (if (> x y) y x))
```

6

(b) What is the output of the following function in Clisp, justify your answer by **tracing** the function call (func 17 3).

```
> (defun func (a b)
    ( if (> b a) 0
     (+ 1 (func (- m n) n ) ) ) )

> (func 17 3) = 5
```

(1 + (func 14 3)) = 5

(1 + (func 11 3)) = 4

(1 + (func 8 3)) = 3

(1 + (func 5 3)) = 2

(1 + (func 2 3)) = 1

0

6

what the function **func** do?

2

function func(m,n) computes & returns the value of m div n (m/n)

(12) **[3]** Given the following simple grammar:

block → **begin** decls stmts **end**
decls → **var** dec-item | λ
dec-item → **D** | **D ,** dec-item
stmts → statement **;** stmts | λ
statement → **S** | λ

V_T ={**begin, end, var, D, S**, ; , , } V_N ={block, decls, decl-item, stmts, statement }

(a) Give a program generated by this grammar.

4

```
begin
  var D , D
  S;
  S;
  S;
end
```

(b) Rewrite production rules using **EBNF** notations.

4

block → begin decls stmts end

decls → var dec-item | λ

dec-item → D{, D}*

stmts → (statement ;)*

statement → S | λ

(c) Compute **FOLLOW**(dec-item).

4

FOLLOW(dec-item) = FOLLOW(decls)
= FIRST(stmts) ∪ {end}
= {S, ;} ∪ {end} = {S, ;, end}

[4] (a) Given the following in C code:

```
int power(int m, int n);      // The function power computes and returns m^n

void main()
{  const int max=10;
   float x=1, y=10;
   x  += y;
   int p = power (max,2);
```

Draw the **symbol table** after the above code is executed.

Name	Type	Value
Power	function Name	~~100~~
max	integer Constant	10
x	float Variable	~~11~~
y	float Variable	10
p	integer Variable	100

(b) Given the grammar:

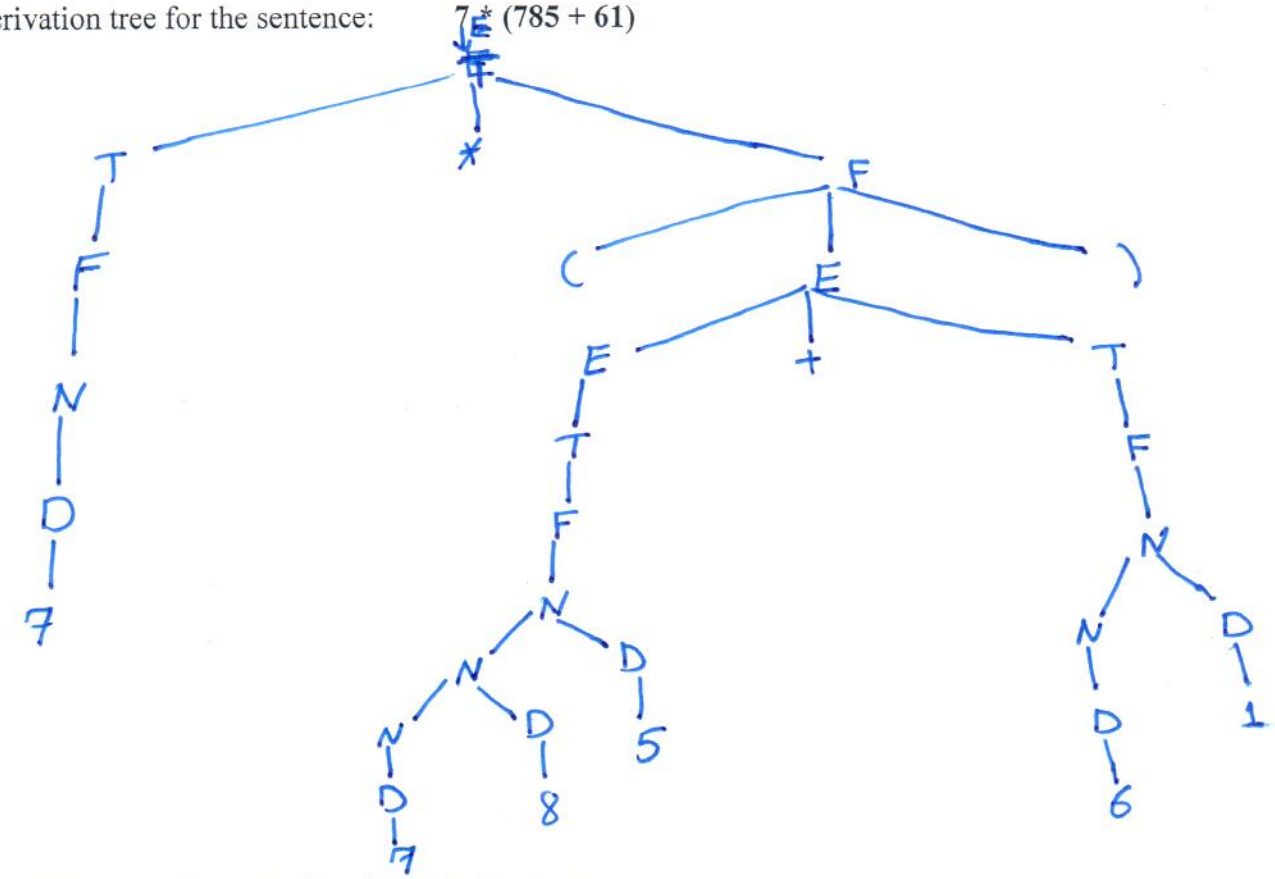
G → S$
S → AS
A → AAB | a | λ
B → bBS | c | λ

	FIRST	FOLLOW
G	a b c ~~$~~	—
S	a b c	~~$~~ a b c
A	a b c λ	a b c
B	b c λ	a b c

[5] **(a)** Given the grammar:

$E \longrightarrow E + T \mid T$
$T \longrightarrow T * F \mid F$
$F \longrightarrow (E) \mid N$
$N \longrightarrow ND$
$N \longrightarrow D$
$D \longrightarrow 0 \mid 1 \mid 2 \mid \ldots \mid 9$

Draw the derivation tree for the sentence: **7 * (785 + 61)**



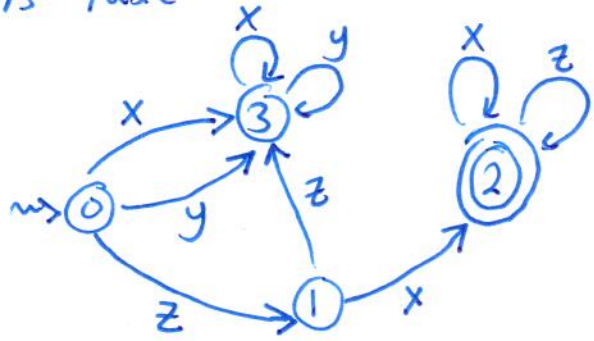
(b) Given the following DFSA. Reduce it to **minimum** states.

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

	x	y	z
(3,4)	(3,4)	(4,3)	
(2,5)	(6,6)		(5,6)
(2,6)	(6,2)		(5,5)
(5,6)	(6,2)		(6,5)

Feasible Pairs Table

$3 \equiv 4$
$2 \equiv 5 \equiv 6$

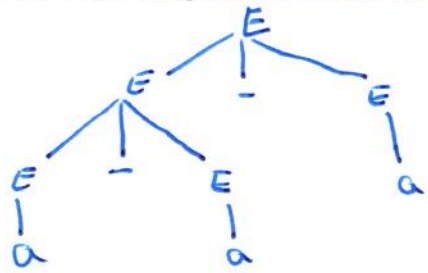


	x	y	z
0	3	3	1
1	2		3
2	2		2
3	3	3	

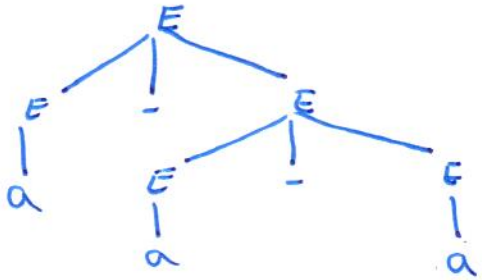
[6] Given the grammar:

$E \rightarrow E - E \mid (E) \mid a$

(a) Show that the grammar is **ambiguous**.



a-a-a



a-a-a

Two derivation trees

(b) A student transform the above grammar to the following none ambiguous grammar:

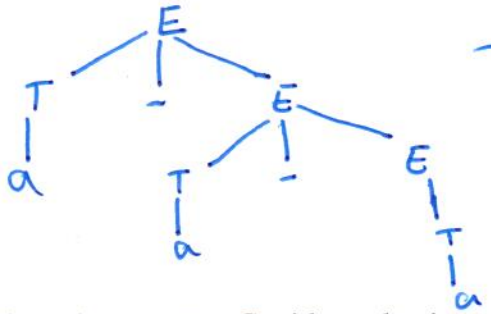
$E \rightarrow T - E \mid T$

$T \rightarrow (E) \mid a$

What is wrong with code. Explain.

This grammar is right associative which contradicts the associative rules in the "-" operations

Consider the sentence a-a-a, its derivation tree is:



This mean that a-a-a will be executed as a-(a-a) which contradict the associativity rule.

(c) Given the grammar G with productions:

$A \rightarrow \alpha$

$A \rightarrow \beta$

$A \rightarrow \lambda$, where $\alpha, \beta \neq \lambda$

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