KEY

100 points
each question 20 point

BIRZEIT UNIVERSITY

College of Engineering & Technology

Computer Science Department

Name:---------- **Number:**----------

Comp439 **Midterm Exam** **Spring 18/19**

(25) **[1] (a)** Explain briefly the meaning of **Machine Readability** of a PL.

5 The ability to write a Translator (compiler or interpreter) for the programming language in an unambiguous way and finite.

(b) Explain the meaning of data abstraction in **Human Readability** of a PL.

Giving abstraction (names) for data types.

5 (a) Simple: such as int, float

(b) Compound: Array data type

(c) The major advantage of the **compiler** is ---It generates object code---,

5

While the major advantage of the **interpreter is** ---Portability---.

(d) The 4 paradigms of programming languages are:

1- Imperative (procedural) Paradigm.

2. Functional :.

5

3. Logical :.

4- Object Oriented :.

[2] (a) Given the following function in CLISP

```
> (defun  func ( n  m)
    ( if  ( = m  1)  n
      ( + ( func  n  ( - m  1)) n )))
```

Trace the function call **(func 3 5).** What do you think **func** do?

> **(func 3 5)** = 15

(func 3 4) + 3 = 15

(func 3 3) + 3 = 12

(func 3 2) + 3 = 9

(func 3 1) + 3 = 6

The function computes n * m

(b) Given the following function in C language:

```
int  doit ( int m,  int n )
{   if ( n == 0 )
        return  m;
    else
        return  doit (n, m % n);
}
```

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

```
>(defun doit (m n)
   (if (= n 0) m
     (doit n (% m n))))
```

The function computes the gcd(m, n)

[3] The production rules for the **exp** structure in LISP given in the **BNF** format looks like:

exp ⟶ **(list)** | **n**

list ⟶ **list , exp** | **exp**

Where : V_N = {**exp , list**} , V_T = {**(,) , , , n**}

(a) Rewrite the above production rules in **EBNF**.

List → List , exp → List , exp , exp → List , exp , exp , exp

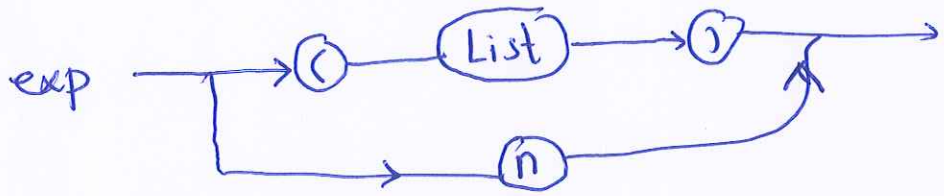
→ exp , exp , exp , ... , exp

∴ exp → (List) | n

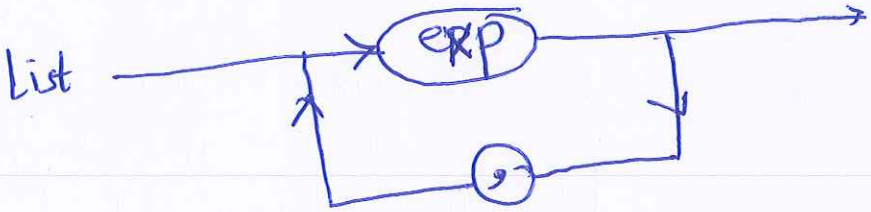
List → exp (, exp)*

(b) Express the EBNF production rules using **syntax diagrams**.

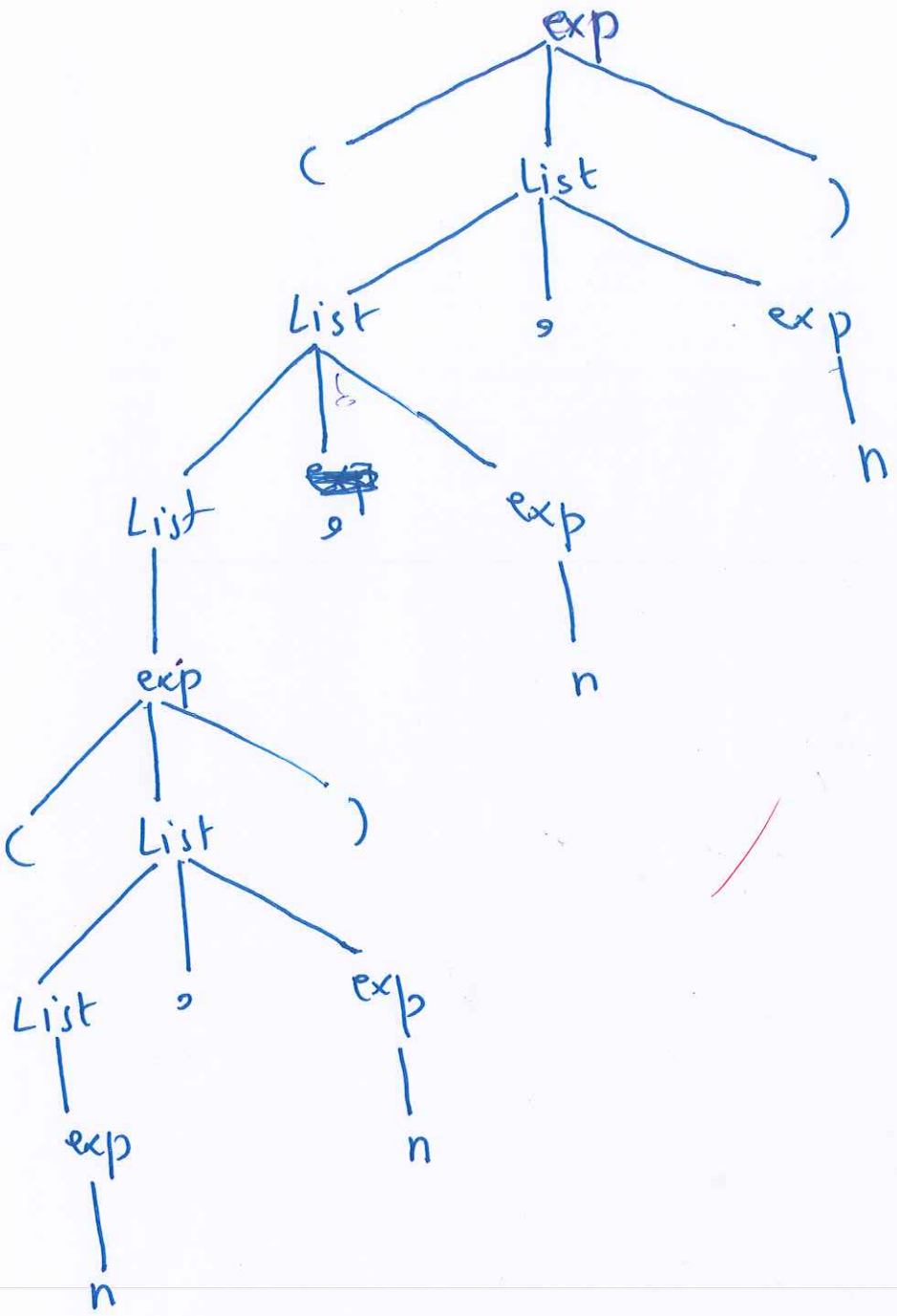
exp :



List

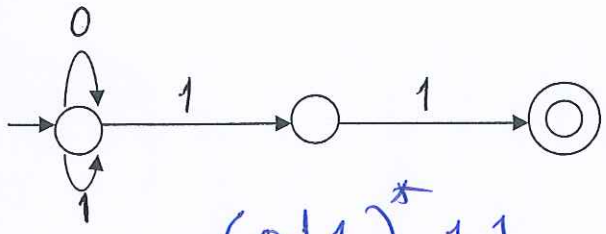


(c) Draw the derivation tree for the sentence **((n, n), n , n)**



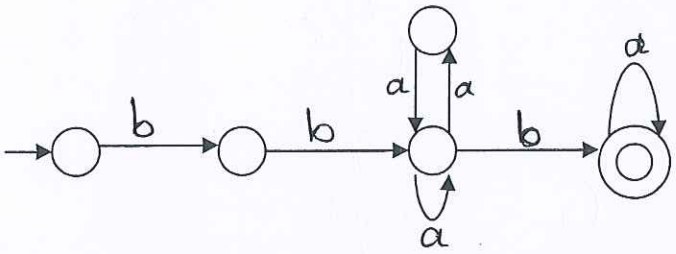
[4] (a) What is the language recognized by the following FSA:

(i)



$(0|1)^* 11$

(ii)



$bb(aa|a)^* ba^*$

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

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

	x	y	z
✓(0,3)	(3,3)	(3,4)	
✓(0,4)	(3,1)	(3,4)	
✓(3,4)	(3,1)	(4,4)	
✓(2,6)	(6,2)	(4,0)	

Feasible-Pairs Table

∴ No Equivalent states.

[5] (a) Given the grammar:

(a)

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

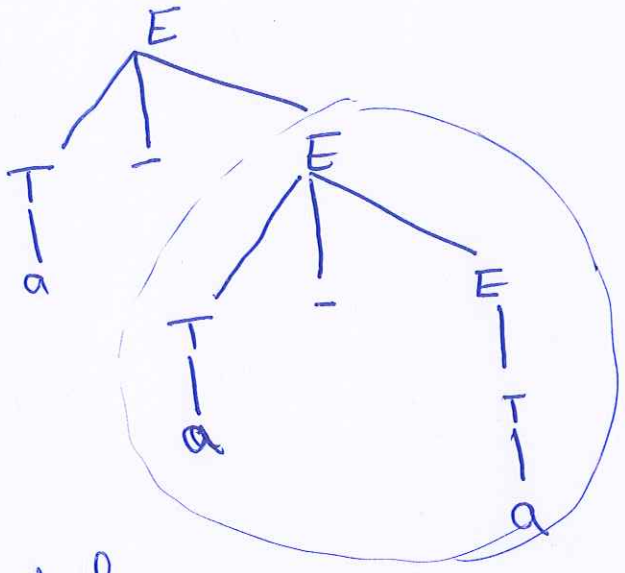
What is the problem with this grammar? Explain your answer in full.

Problem is the right associative grammar.

Because in this grammar, a - a - a means a - (a - a) which contradicts our associativity protocol.

Let us derive a - a - a by showing the derivation tree

(b)



which means that

a - a - a ≡ a - (a - a) because the subtree is evaluated first.

(b) The grammar for the **if…else…** structure can be written as:

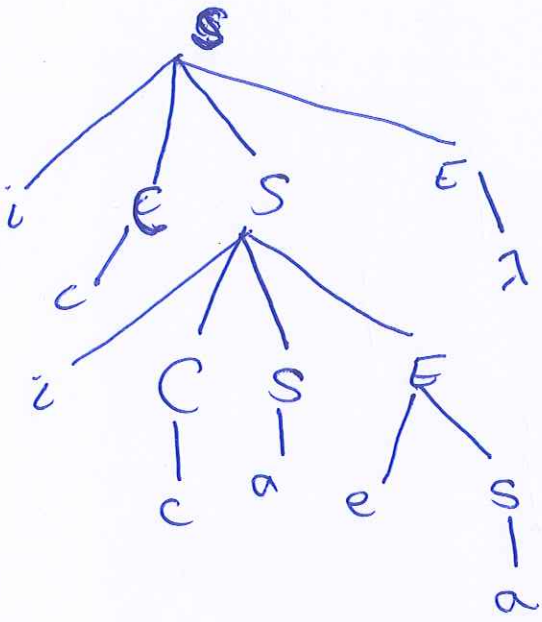
$S \rightarrow iCSE \mid a$
$E \rightarrow eS \mid \lambda$
$C \rightarrow c$

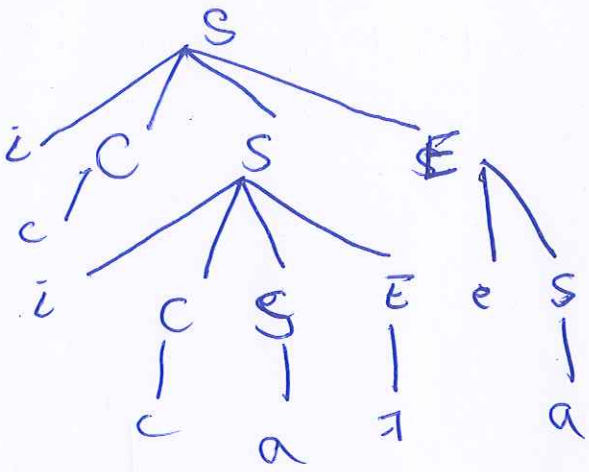
$V_N = \{ S, C, E\}$, $V_T = \{ i, a, c, e\}$

(a) Derive the sentence **i c i c a e a.** Show that the above grammar is ambiguous.

$S \rightarrow iCSE \rightarrow icSE \rightarrow iciCSEE \rightarrow icicSEE \rightarrow icicaEE$
$\rightarrow icicaeSE \rightarrow icicaeaE \rightarrow icicaea$ ($E \rightarrow \lambda$)

10





icicaes

icicaea

Two derivation Trees for the same sentence
it is ambiguous