

1. (30) True or False. Use grammars A, B, C in the following.

A. $\langle Q \rangle ::= \langle Q \rangle ? \langle Q \rangle \mid \langle R \rangle$
 $\langle R \rangle ::= \langle R \rangle ! \langle R \rangle \mid (\langle Q \rangle) \mid \mathbf{a} \mid \mathbf{b} \mid \mathbf{c}$

B. $\langle L \rangle ::= \langle M \rangle @ \langle L \rangle \mid \langle M \rangle$
 $\langle M \rangle ::= \langle M \rangle \% \langle N \rangle \mid \langle N \rangle$
 $\langle N \rangle ::= (\langle M \rangle) \mid \mathbf{x} \mid \mathbf{y} \mid \mathbf{z}$

C. **<!ELEMENT x (a+,b*)>**
<!ELEMENT y ((y,a) | a) >
<!ELEMENT a EMPTY>
<!ELEMENT b EMPTY>

- a. T Grammar B defines the precedence of @ below %.
- b. T Grammar B defines @ as left-associative.
- c. T By grammar A "(a ? b) ! c" and "a ? b ! c" parse to the same AST.
- d. F By B, "x @ y @ z" and "x @ (y @ z)" parse to the same AST.
- e. F By B, "x % (y % z)" and "x % y % z" parse to the same AST.
- f. T By A defines the precedence of ? above !.
- g. T By A, "(a ? (a ? a))" is valid.
- h. T Grammar A is ambiguous, allows multiple valid parses.
- i. F Grammar B is ambiguous, allows multiple valid parses.
- j. F By C, XML "<x></x>" is valid.
- k. T By C, XML "<y><a/></y>" is valid.
- l. T By C, XML "<y><a/><y><a/><y><a/></y></y></y>" is valid.
- m. T By C, XML "<y><y><y><a/></y><a/></y><a/></y>" is valid.
- n. F By C, XML "<x></x>" is valid.
- o. F By C, XML "<x><a/><a/><a/></x>" is valid.

2. (8) Give a complete grammar, in BNF or EBNF, for the following languages:

a. All even binary numbers (i.e. end with 0). Examples: 0101010, 0, 111110

```
<digit> ::= <zero> | <one> | <digit>
<zero> ::= 0
<one> ::= 1
<even-binary> ::= (<digit>)+ <zero> | <zero>
```

b. All binary numbers of at least three consecutive 1's and three consecutive 0's. The language includes 111000 and 110001011101 but not 0011000101011.

Multiple Repitions of Consecutive numbers:

```
<consecutive-binary> ::= [<binary>+] (<one> <one> <one>) [<zero>] [<binary>+] |
[<binary>+] (<zero><zero><zero>) [<one>] [<binary>+]
<binary> ::= <zero> | <one> | <zero>+ | <one>+
<zero> ::= 0
<one> ::= 1
```

3. (10) Syntax of the *ADMIN* language is quite simple yet only administrators can speak a complete <sentence> without making mistakes. The alphabet of the language is {T, L, A, _} where _ stands for a space. The grammar is:

```
<stop> ::= L | A
<plosive> ::= <stop> A
<syllable> ::= <plosive> | <plosive> <stop> | T <plosive> | T <stop>
<word> ::= <syllable> | <word> <syllable>
<sentence> ::= <word> | <sentence>_<word>
```

Which of the following speakers is posing as an administrator (i.e. cannot correctly speak a <sentence> of TLA language)? Mark each as an administrator or not.

a) SAY_WHAT not

b) TLA administrator

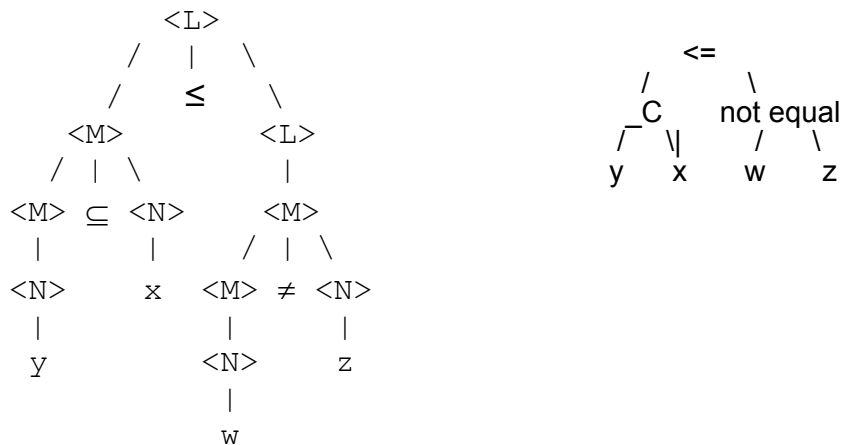
c) T not

d) TLATLA not

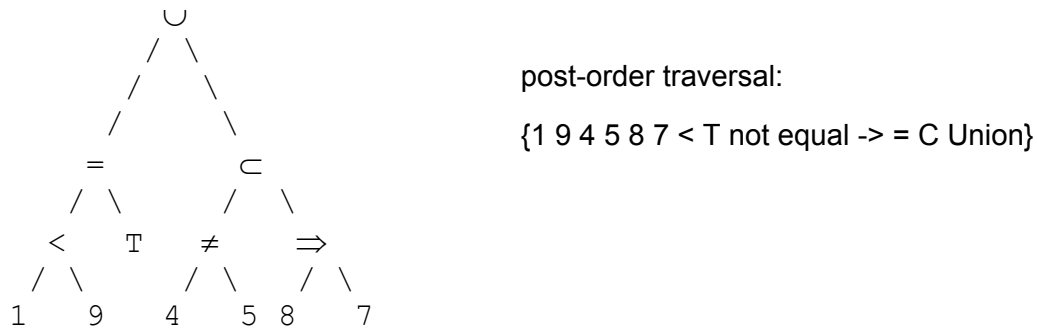
e) TLA_TLA_TLA administrator

4. (15)

a. Give the corresponding AST for the following.



b. Give the post-order traversal (i.e. RPN) of the following AST.



c. Represent the RPN at right as standard, infix expression: $1\ 2\ +\ 3\ *\ 8\ 4\ 7\ -\ +\ /$

-5.6667

$1+2 = 3\ 3 * 3 = 9 \rightarrow 9 + 8 = 17$

$17 / -3 = -5.666$

$4 - 7 \rightarrow -3$

$[(((1+2) * 3) + 8) / (4 - 7)]$

d. Evaluate the RPN: $1\ 2\ 3\ *\ +\ 8\ 4\ /\ +\ 7\ -$

$[(2+3) + (8 / 4)] - 7$
 $[5 + 2] - 7 = 0$

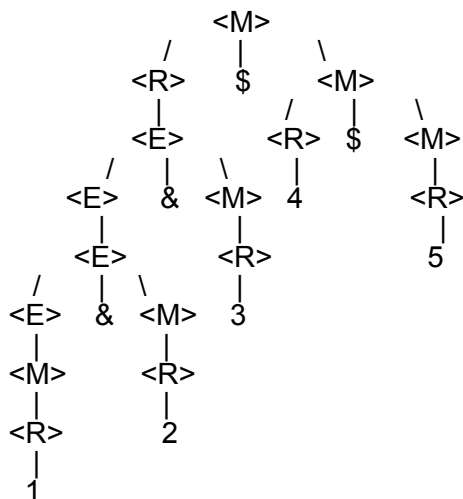
Resulting in 0

$[((2+3) + (8 / 4)) - 7]$

5. (12) Use the following grammar

$\langle e \rangle ::= \langle e \rangle \& \langle m \rangle \mid \langle m \rangle$
 $\langle m \rangle ::= \langle r \rangle \$ \langle m \rangle \mid \langle r \rangle$
 $\langle r \rangle ::= (\langle e \rangle) \mid 1 \mid 2 \mid 3 \mid 4 \mid 5$

a. Give the parse tree for: **(1 & 2 & 3) \$ 4 \$ 5**



b. Modify the following grammar to add a left-associative rule for a binary operator \perp with precedence between that of \otimes and \oplus .

$\langle r \rangle ::= \langle r \rangle \otimes \langle s \rangle \mid \langle s \rangle$

$\langle s \rangle ::= \langle t \rangle \oplus \langle s \rangle \mid \langle t \rangle$

$\langle t \rangle ::= (\langle r \rangle) \mid 1 \mid 2 \mid 3 \mid 4 \mid 5$

$\langle r \rangle ::= \langle r \rangle \text{ OX } \langle s \rangle \mid \langle s \rangle$

$\langle u \rangle ::= \langle t \rangle _ _ \langle u \rangle \mid \langle t \rangle$

$\langle s \rangle ::= \langle t \rangle \text{ O+ } \langle s \rangle \mid \langle t \rangle$

$\langle t \rangle ::= (\langle r \rangle) \mid 1 \mid 2 \mid 3 \mid 4 \mid 5$

6. (15) STATIC ACTIVATION RECORDS: Fill in the two tables with activation record contents, label and the value through the complete program execution. Use call-by-reference parameter passing and static activation records.

```

1. void asub(double x[], double &r)
2. {
3.     double I;
4.     bsub(x, I);
5.     r = I;
6. }
7.
8. void bsub(double t[], double &s)
9. {
10.    s = t[0]+t[1]+t[2];
11. }
12.
13. void main(void )
14. {
15.    double y[3];
16.    y[0] = 4.0;
17.    y[1] = 5.0;
18.    y[2] = 1.0;
19.    double a = 2.0;
20.    asub(y, a);
21.    cout << a;
22. }

```

Table 2 – Values

Address	Symbol	Value
123	y	
124		
125		
126	a	2.0
127	i	

Table 1 – Activation Records

Addr.	Contents	Label	
345		IP	AR main
346	OS	DL	
347	registers	TMP	
348		IP	AR asub
349		PAR[1]	
350		PAR[2]	
351		DL	
352		TMP	
353		IP	AR bsub
354		PAR[1]	
355		PAR[2]	
356		DL	
357		TMP	
358			
359			
360			
361			
362			
363			
364			


```

type AST =
| Const of int
| Var of string
| Plus of AST * AST
| Times of AST * AST
| Let of string * AST * AST
| Fn of string * AST
| Apply of AST * AST;;

let rec lookup V L =
  let (var,value)::T = L
  if V=var then value else lookup V T;;

let rec valN exp context =
  match exp with
  | Const x -> Const x
  | Var V -> lookup V context
  | Plus (Const x, Const y) -> Const (x+y)
  | Plus (x, y) -> valN (Plus(valN x context, valN y context)) context
  | Times (Const x, Const y) -> Const (x*y)
  | Times (x, y) -> valN (Times(valN x context, valN y context)) context
  | Let (V, exp1, exp2) -> valN exp2 ((V, valN exp1 context)::context)
  | Fn (formal, body) -> Fn(formal, body)
  | Apply (Fn(formal, body), actual) ->
    valN body ((formal, valN actual context)::context)
  | Apply (Var v, actual) ->
    valN (Apply (valN (Var v) context, actual)) context;;

```