# CMSC330 Spring 2024 Quiz 3 Solutions

## Problem 1: Context Free Grammars - Derivations

[Total 8 pts]

Consider the following Grammar:

**Version A and C:**

```
S -> if S then S T | U
T -> else S | ϵ
U -> ‘hello’ | ‘bye’ | true | false
```

**Version B and D:**

```
S -> if S then S T | U
T -> else S | ϵ
U -> n | true | false (∗ where n is a positive integer ∗)
```

(a) Is this an ambiguous grammar? (For all versions) [2 pts]

[✓ A Yes]   B No

(b) Derive: [6 pts]

**Version A:** "if if true then ‘hello’ else ‘bye’ then false"

```
S -> if S then S T -> if if S then S T then S T ->
   if if true then S T then S T -> if if true then U T then S T ->
   if if true then ’hello’ T then S T -> if if true then ’hello’ else S then S T ->
   if if true then ’hello’ else U then S T -> if if true then ’hello’ else ’bye’ then S T ->
   if if true then ’hello’ else ’bye’ then U T -> if if true then ’hello’ else ’bye’ then false T ->
   if if true then ’hello’ else ’bye’ then false
```

**Version B:** "if 19 then if true then 21 else false"

```
S -> if S then S T -> if U then  S T -> if 19 then S T ->
   if 19 then if S then S T T -> if 19 then if U then S T T ->
   if 19 then if U then S T T -> if 19 then if true then S T T ->
   if 19 then if true then U T T -> if 19 then if true then 21 T T ->
   if 19 then if true then 21 else S T -> if 19 then if true then 21 else U T ->
   if 19 then if true then 21 else false T -> if 19 then if true then 21 else false
```

**Version C:** "if true then if false then ‘hello’ else ‘bye’"

```
S -> if S then S T -> if U then S T -> if true then S T ->
   if true then if S then S T T -> if true then if U then S T T ->
   if true then if false then S T T -> if true then if false then U T T ->
   if true then if false then ’hello’ T T -> if true then if false then ’hello’ else S T ->
   if true then if false then ’hello’ else U T -> if true then if false then ’hello’ else ’bye’ T ->
   if true then if false then ’hello’ else ’bye’
```

**Version D:** "if if true then false else 13 then true"

```
S -> if S then S T -> if if S then S T then S T -> if if U then S T then S T ->
   if if true then S T then S T -> if if true then U T then S T ->
   if if true then false T then S T -> if if true then false else S then S T ->
   if if true then false else U then S T -> if if true then false else 13 then S T ->
   if if true then false else 13 then U T -> if if true then false else 13 then true T ->
   if if true then false else 13 then true
```

## Problem 2: Context Free Grammars - Creation

[Total 4 pts]

(a) Design a CFG that represents the same set of strings as the regular expression: [4 pts]

**Version A:** (d|e)*f+

```
S -> AB
A -> dA | eA | ϵ
B -> fB | f
```

**Version B:** e+(a|d)*

```
S -> AB
A -> eA | e
B -> aB | dB | ϵ
```

**Version C:** a*(b|c)+

```
S -> AB
A -> aA | ϵ
B -> bB | cB | b | c
```

**Version D:** (b|d)+f*

```
S -> AB
A -> bA | dA | b | d
B -> fB | ϵ
```

## Problem 3: Lexing Parsing and Evaluating

[Total 8 pts]

Given the following CFG, and assuming the **Ocaml** type system, at what stage of language processing would each expression **fail**? Mark **'Valid'** if the expression would be accepted by the grammar and evaluate properly. Assume the only symbols allowed are those found in the grammar.

$$E \rightarrow M \text{ and } E \mid M \text{ or } E \mid M$$
$$M \rightarrow N + M \mid N - M \mid N$$
$$N \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid \texttt{true} \mid \texttt{false} \mid (E)$$

| | Lexer | Parser | Evaluator | Valid |
|---|---|---|---|---|
| `1 + 2 - (true and false)` | L | P | [✓ E] | V |
| `true + (3 - 2}` | [✓ L] | P | E | V |
| `3 * 1 - 2` | [✓ L] | P | E | V |
| `)(2 or + -` | L | [✓ P] | E | V |
| `true` | L | P | E | [✓ V] |
| `true and (false)` | L | P | E | [✓ V] |
| `(1) + (4)` | L | P | E | [✓ V] |
| `(((5)))` | [✓ L] | P | E | V |
| `3 + 4 - (false or true)` | L | P | [✓ E] | V |
| `{2}` | [✓ L] | P | E | V |
| `4 - 2 * 1` | [✓ L] | P | E | V |
| `2 and 5` | [✓ L] | P | E | V |
| `2 or ) + (` | L | [✓ P] | E | V |
| `false` | L | P | E | [✓ V] |
| `true and (3 + 1 = 4)` | [✓ L] | P | E | V |
| `(({2}))` | [✓ L] | P | E | V |