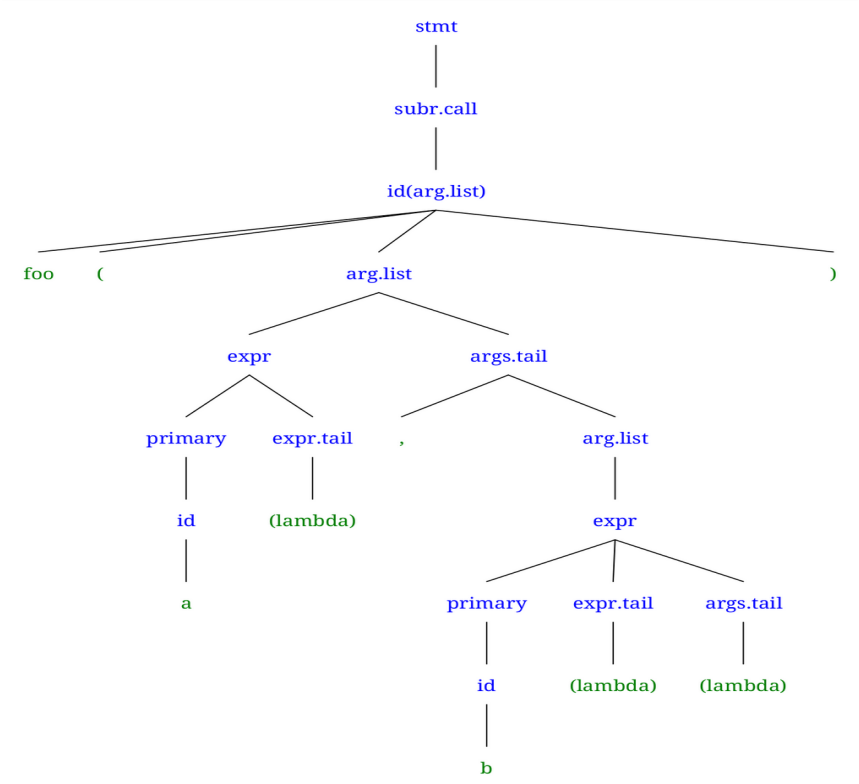
Homework 2

CSC 310 | Peter Schaefer | February 14th 2024

2.13 Consider the following grammar:

```
stmt → assignment | subr_call
assignment → id := expr
subr_call → id (arg_list)
expr  → primary expr_tail
expr_tail → op expr | ε
primary → id | subr_call | ( expr )
op    → + | - | * | /
arg_list → expr args_tail
args_tail → , arg_list | ε
```

a. Construct a parse tree for the input string `foo(a, b)`.



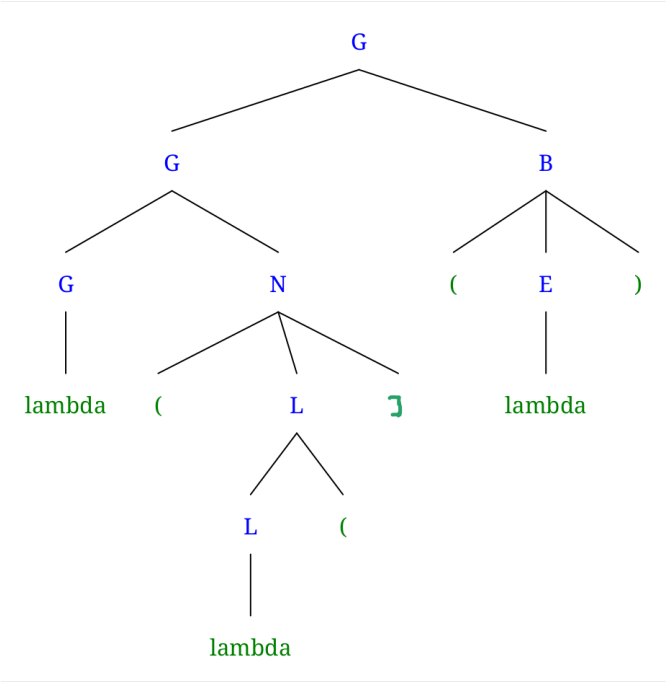
b. Give a canonical (right-most) derivation of this same string.

```
stmt → subr_call → id(arg_list) → id(expr args_tail) → id(expr, arg_list) →
id(expr, expr args_tail) → id(expr, expr ε) → id(expr, primary expr_tail ε) →
id(expr, primary εε) → id(expr, id εε) → id(expr, b εε) → id(primary expr_tail, b εε)
→ id(primary ε, b εε) → id(id ε, b εε) → id(a ε, b εε) → foo(a ε, b εε) = foo(a, b)
```

2.15 Consider the following grammar.

```
G → GB | GN | ε
B → (E)
E → E(E) | ε
N → (L]
L → LE | L( | ε
```

b. Give a parse tree for the string `(()(`.



c. Give a canonical (right-most) derivation of this same string.

`G → GB → G(E) → G(ε) → GN(ε) → G(L](ε) → G(L(](ε) → G(ε](ε) → ε(ε](ε) = (()(`