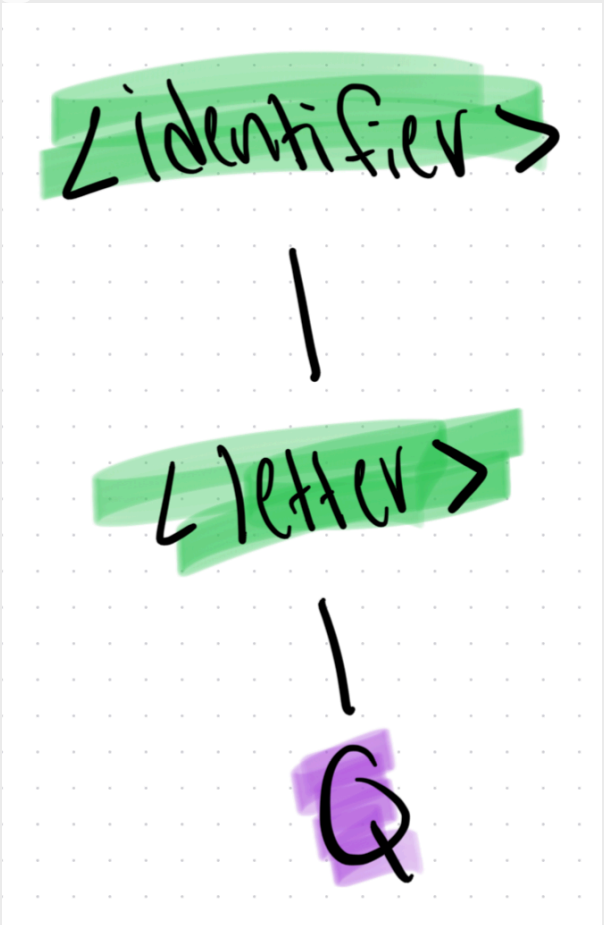
# Homework One

## Question 1

> Which of the following are valid identifiers? For each valid identifier, provide a parse tree (with the `id` abstraction at the root). If invalid, briefly describe what aspect violates the rules.

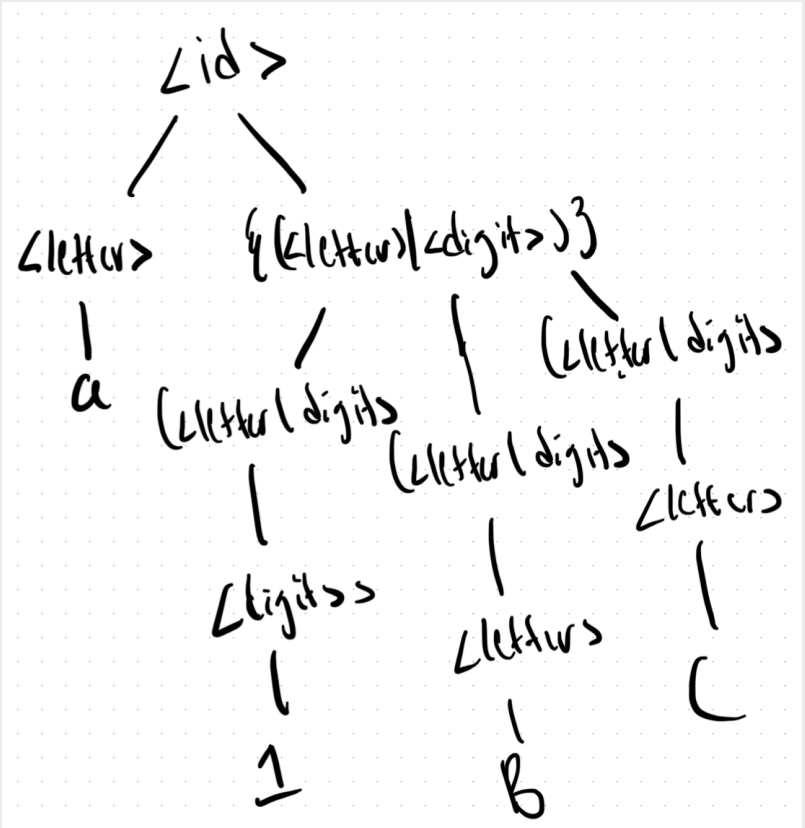
`Q` is valid, shown by the parse tree below:




Parse Tree for `Q`

`4b` is invalid, since it starts with a number.

`a1BC` is valid, shown by the parse tree below:



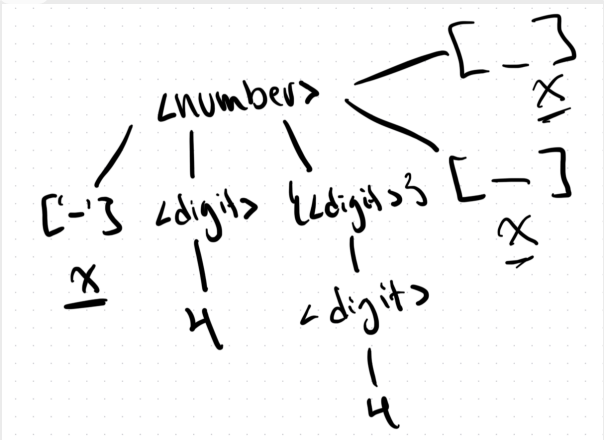

Parse tree for `a1BC`

`X_1` is invalid, since `_` is not a letter or digit.

## Question 2

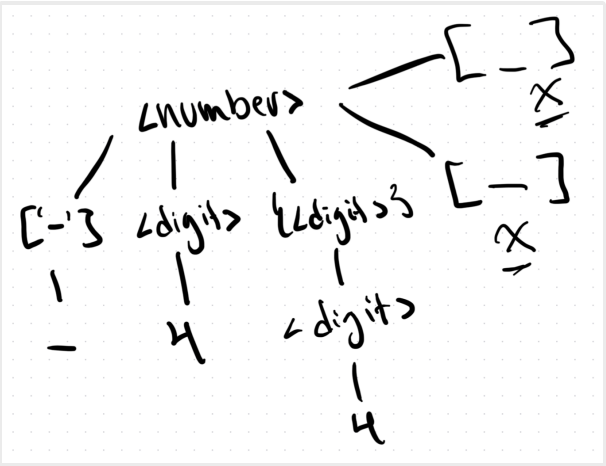
> Which of the following are valid numbers? For each valid number, provide a parse tree (with the number abstraction at the root). If invalid, briefly describe what aspect violates the rules.

`44` is a valid number, shown by the parse tree below:



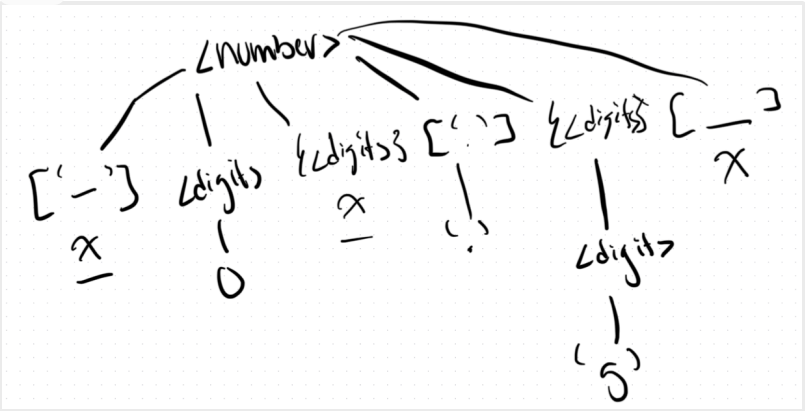

Parse tree for `44`

`-44` is a valid number, shown by the parse tree below:




Parse tree for `44`

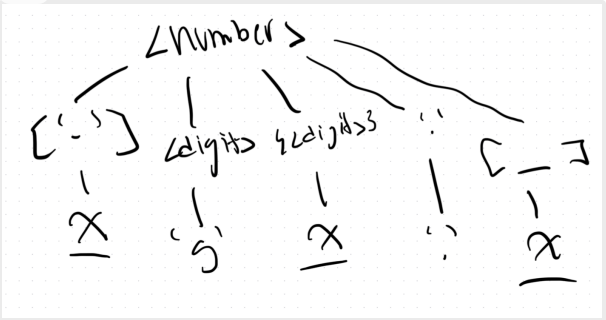
`0.5` is a valid number, shown by the parse tree below:




Parse tree for `0.5`

`.5` is an invalid number, since there needs to be a number preceding the decimal point.

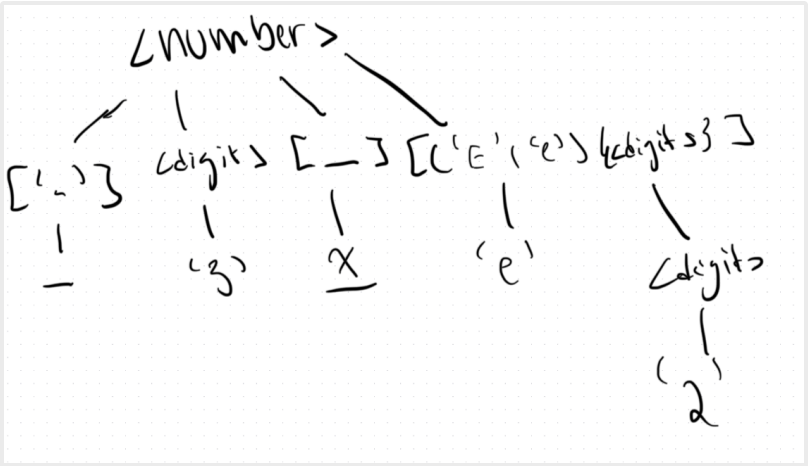
`5.` is valid, shown by the parse tree below:




Parse tree for `5.`

`-.2` is invalid, since there needs to be an actual number preceding the decimal point.
`--9` is invalid, since the preceding `-` is not repetitive.
`E4` is invalid, since there needs to be a preceding number before the `E`.

`-3e2` is valid, shown by the parse tree below:




Proof for `3e2`

`-3.1e2.5` is invalid, since numbers after `e/E` need to be whole.

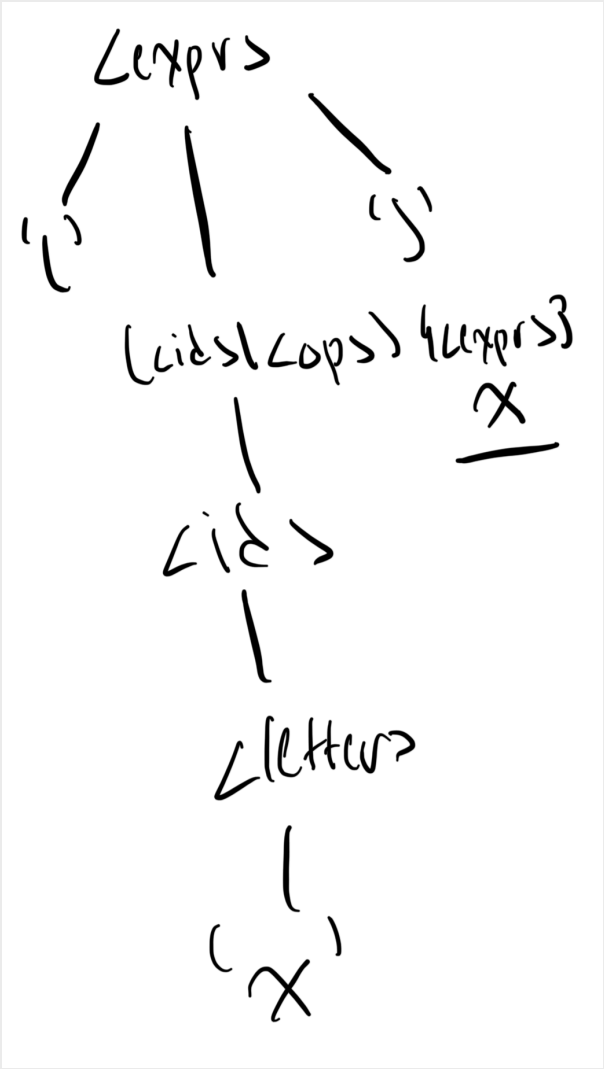
## Question 3

> Which of the following are valid expressions? For each valid expression, provide a parse tree (with the `expr` abstraction at the root). If invalid, briefly describe what aspect violates the rules.

`x` is valid, shown by the parse tree below:


Parse tree for `x`

`(x)` is valid, shown by the parse tree below:

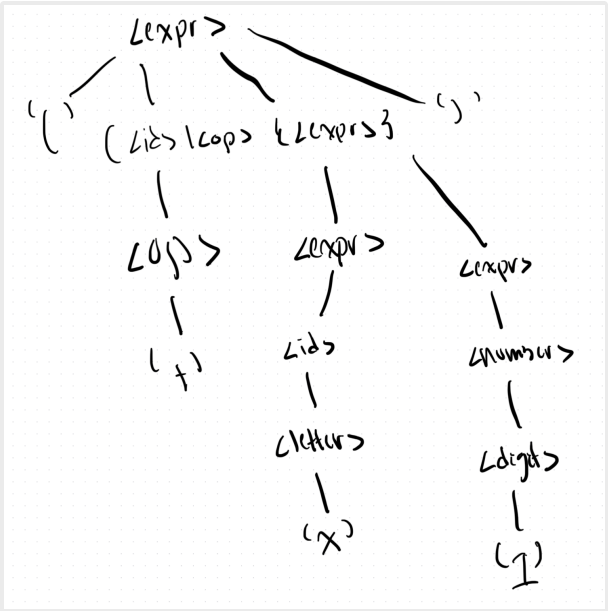



Parse tree for `(x)`

`(-x)` is invalid, because `-x` is not a valid `<id>` or `<op>`.

`(x + 1)` is invalid, because `+` is not an expression.

`(+ x 1)` is valid, shown by the parse tree below:




Parse tree for `(x + 1)`

`(- x 1)` is invalid, since `-` is not an operator

## Question 4

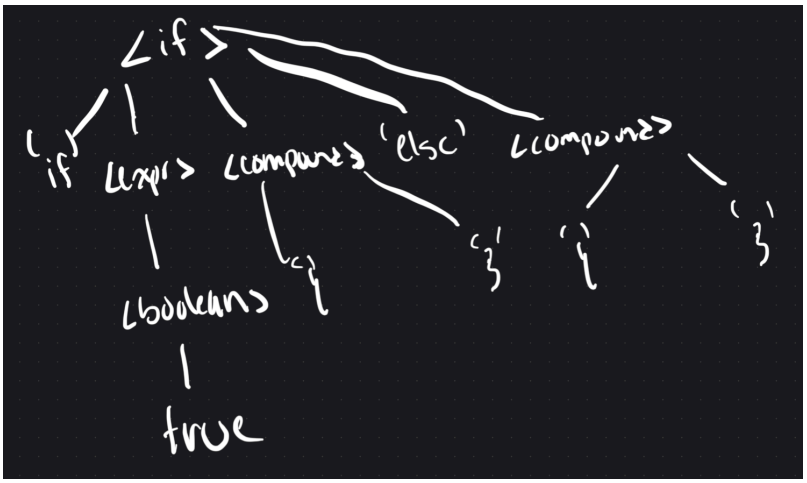
> Which of the following are valid if statements? If valid, provide a parse tree (with the if abstraction at the root). If invalid, briefly describe what aspect violates the rules.

```
if (== x y)
{
        print "eq"
}
```

is invalid since each if statement must have an else.

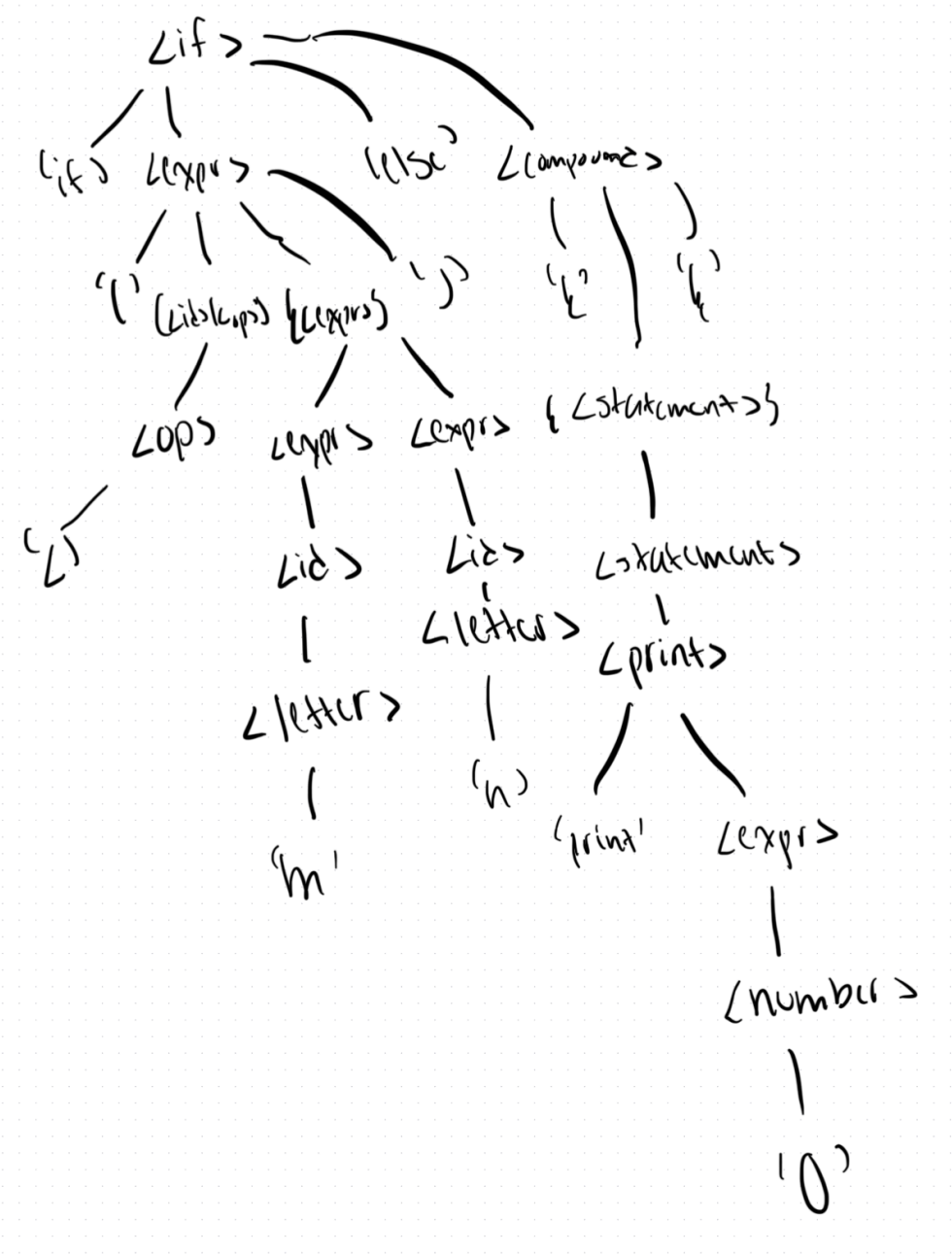
```
if true {}
else {}
```

is valid, shown by the following parse tree:



```
if (> m n)
{
        print 0
}
else
{
        print 1
}
```

is valid, shown by the parse tree below:



```
if (avg >= 80)
{
        if (avg >= 90)
                print "A"
```

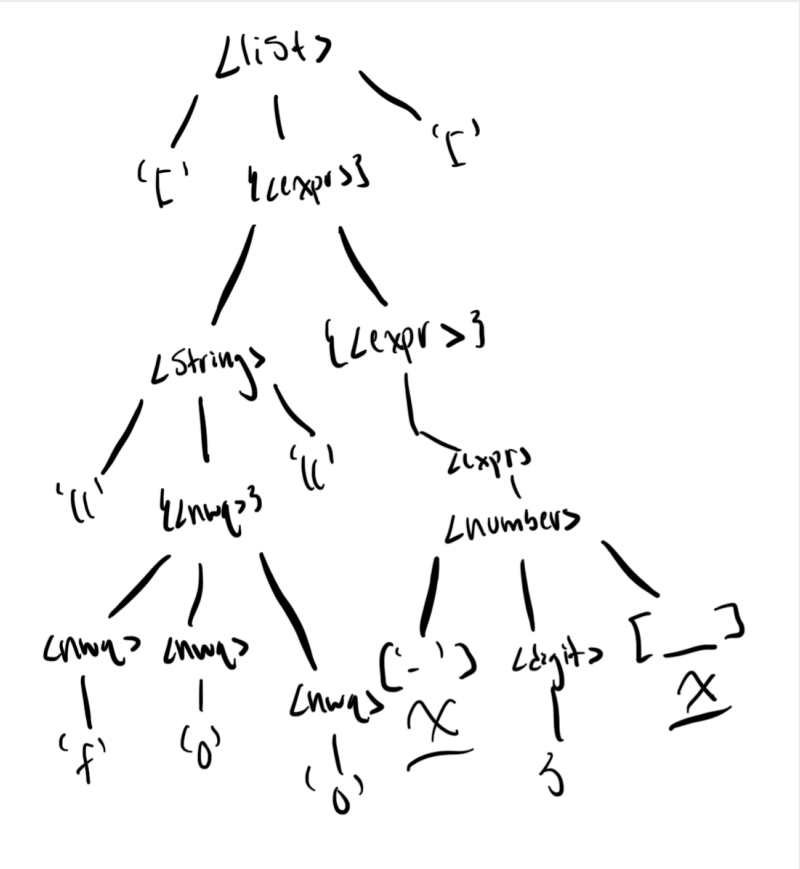
```
        else
                print "B"
}
else
{
        print "F"
}
```

is invalid because there must be curly braces after the if statement

# Question 5

> Is it valid to mix the types of values in a list, e.g., ["foo" 3]? If so provide a parse tree for this list expression. If not, explain why not.

Mixing the types of a list is valid, shown by the the parse tree below:

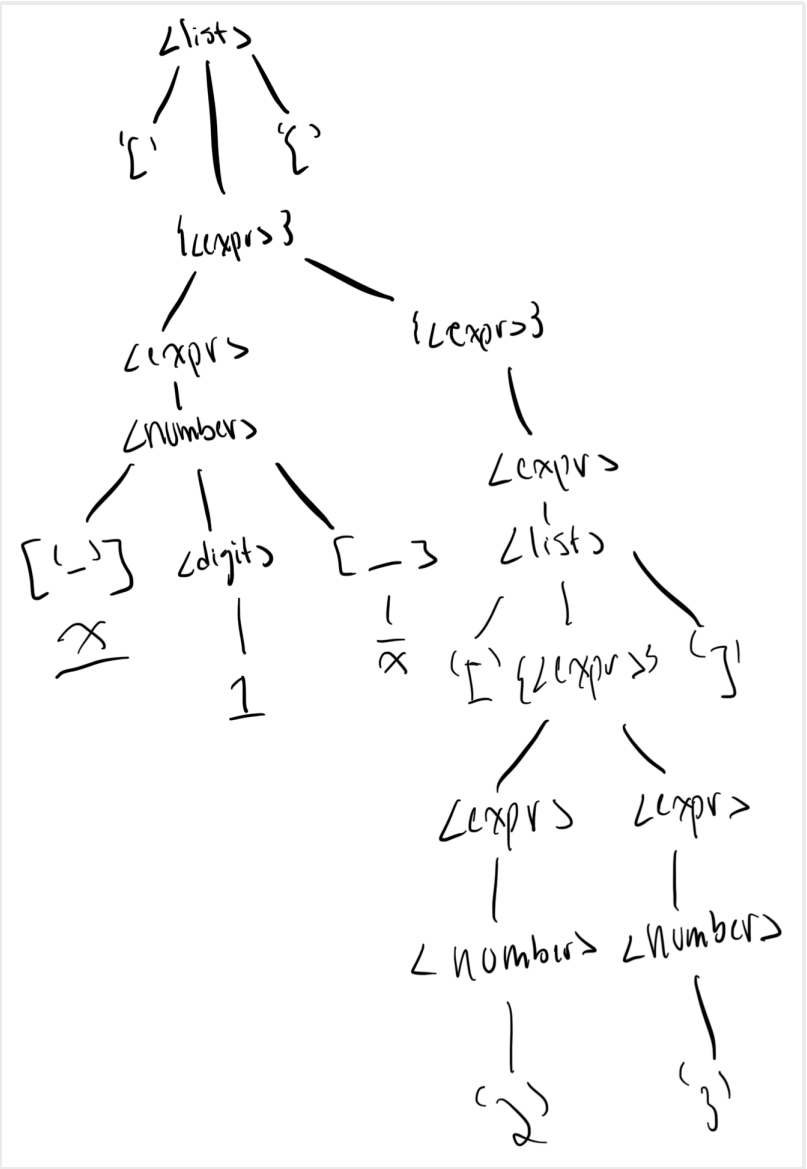



Parse tree for mixed tree

# Question 6

> Is it valid to nest a list inside another list, e.g., [1 [2 3]]? If so provide a parse tree for this list expression. If not, explain why not.

Nesting a list within a list is valid, shown by the parse tree below:




Parse tree for nested tree

## Question 7

```
a = 0
```

## Question 8

```
while 0{}
```

## Question 9

```
func a(){}
```