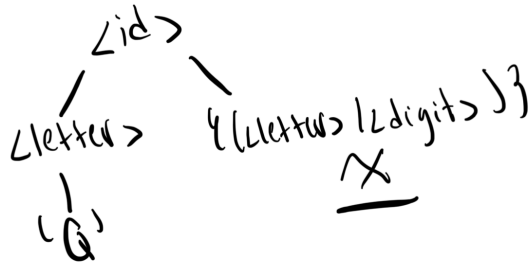


# Homework One

## 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.

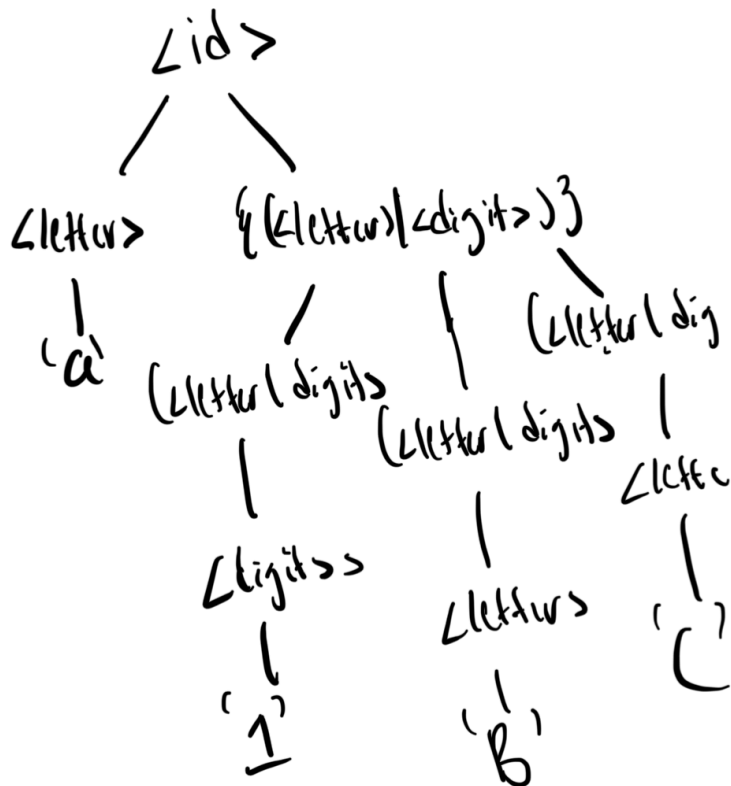
`Q` is valid, shown by the parse tree below:



Parse tree for `Q`

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

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



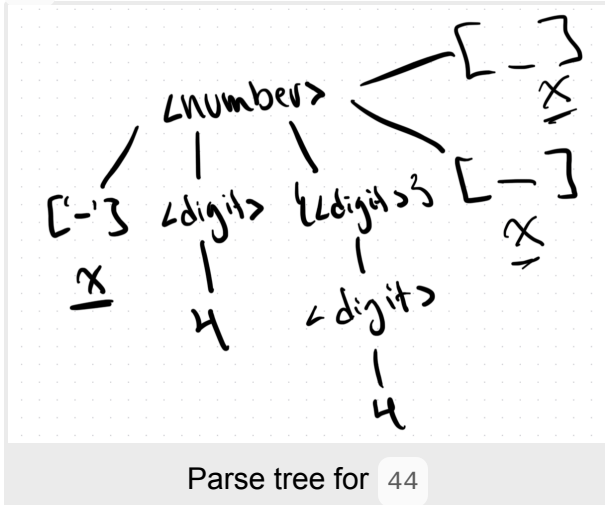
Parse tree for `a1BC`

x\_1 is invalid, since \_ is not a letter or digit.

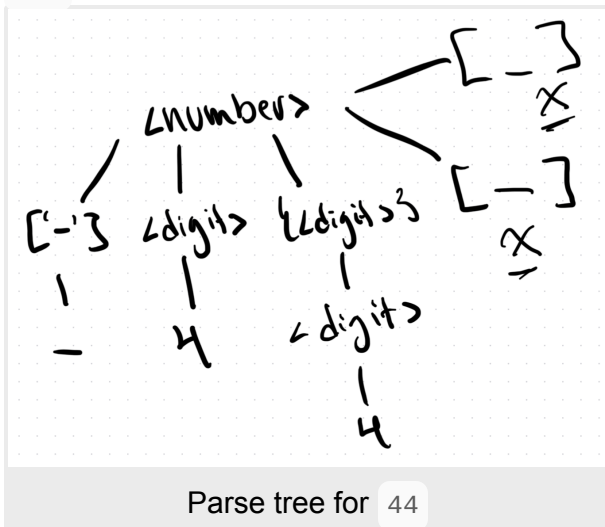
## Question 2

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.

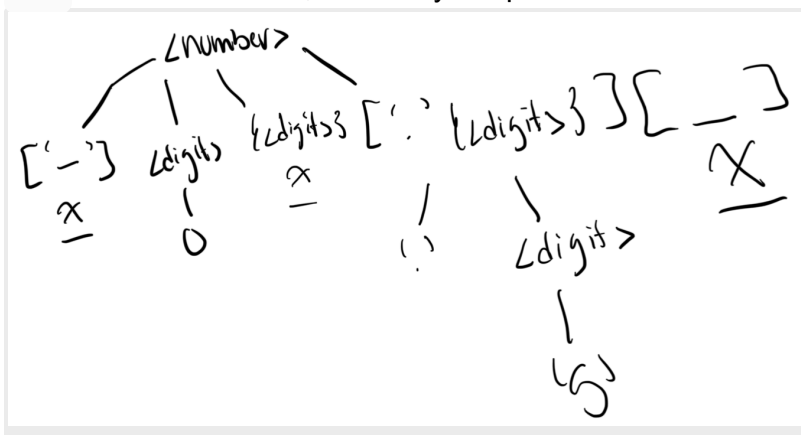
44 is a valid number, shown by the parse tree below:



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



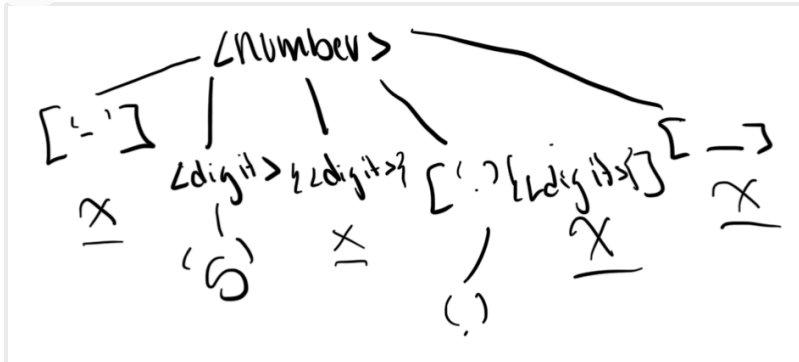
0.5 is a valid number, shown by the parse tree below:



### 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:



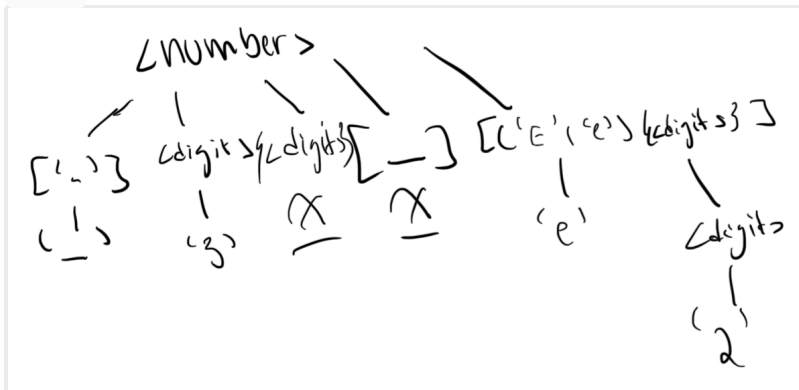
### Parse tree for 5.

- .2 is invalid, since there needs to be a number preceding the decimal point.

--9 is invalid, since the preceding - is not repetitive.

E4 is invalid, since there needs to be a number preceding the E.

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



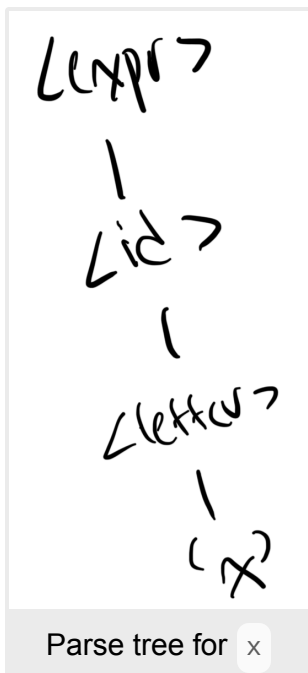
### Parse tree for -3e2

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

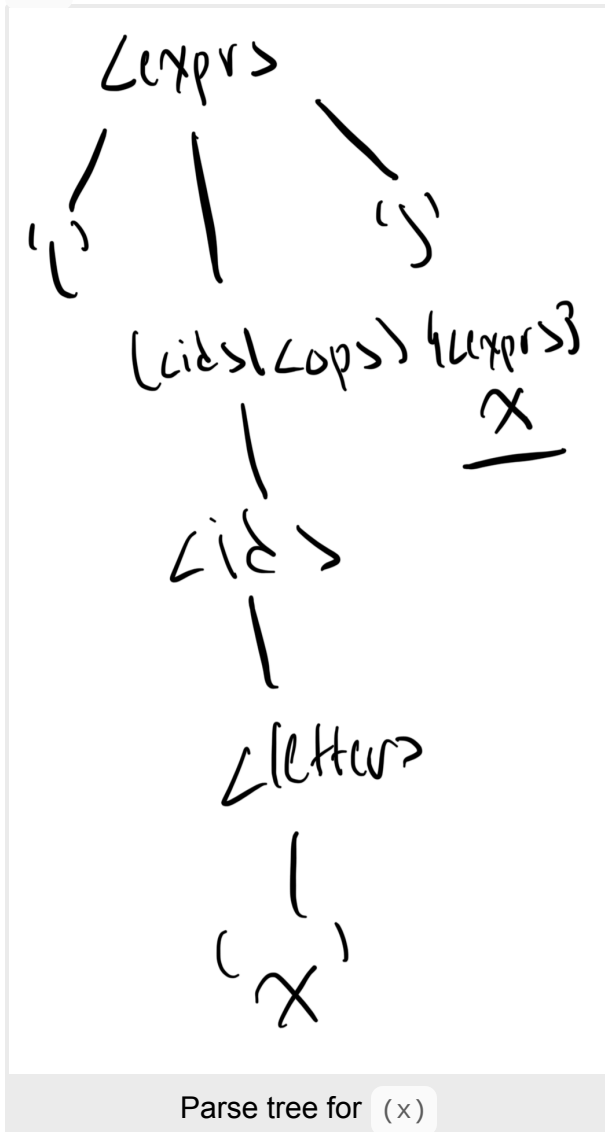
## Question 3

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:



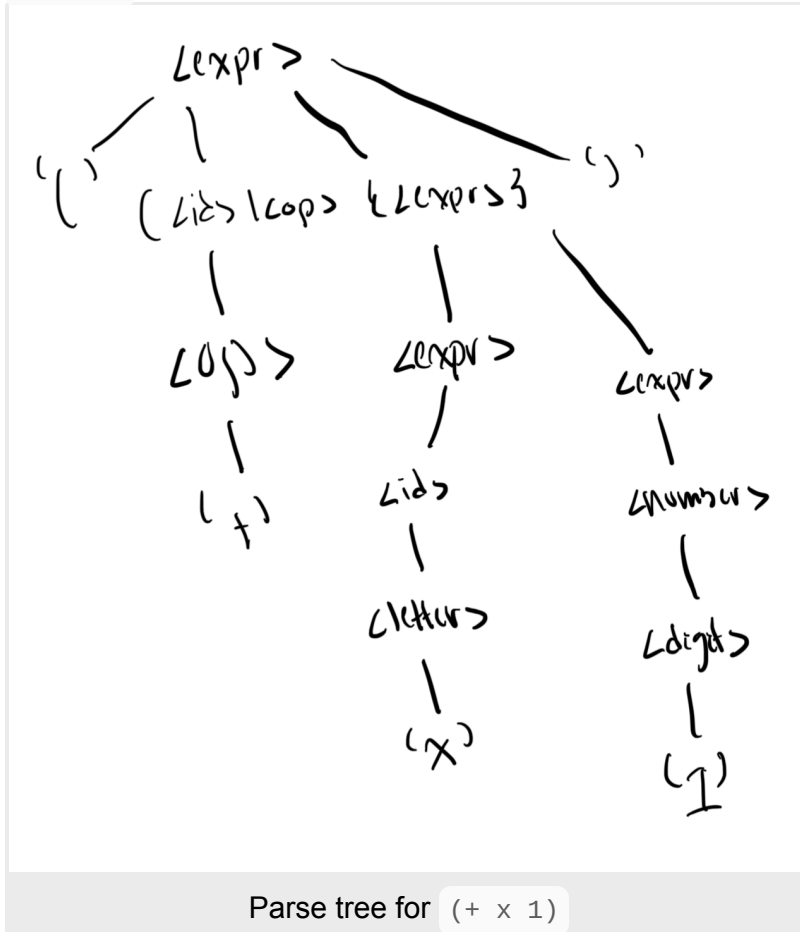
`(x)` is valid, shown by the parse tree below:



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

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

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



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

## Question 4

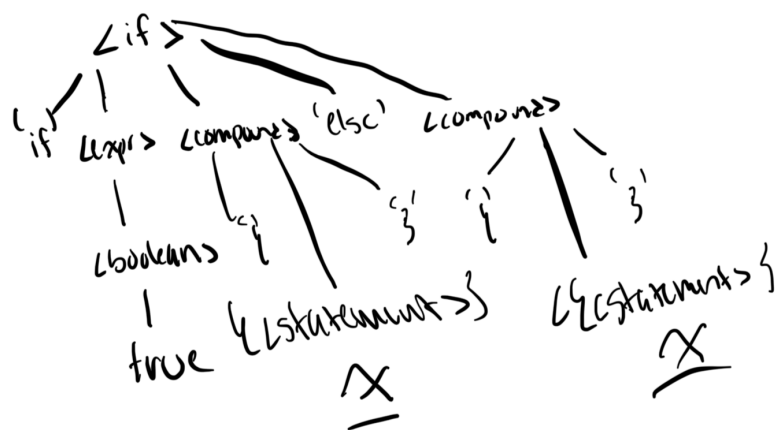
Which of the following are valid if statements? If valid, provide a parse tree (with the if abstraction at the root). If invalid, briefly describe what aspect violates the rules.

```
1  if (== x y)
2  {
3      print "eq"
4  }
```

is invalid since each `if` statement must have an `else`.

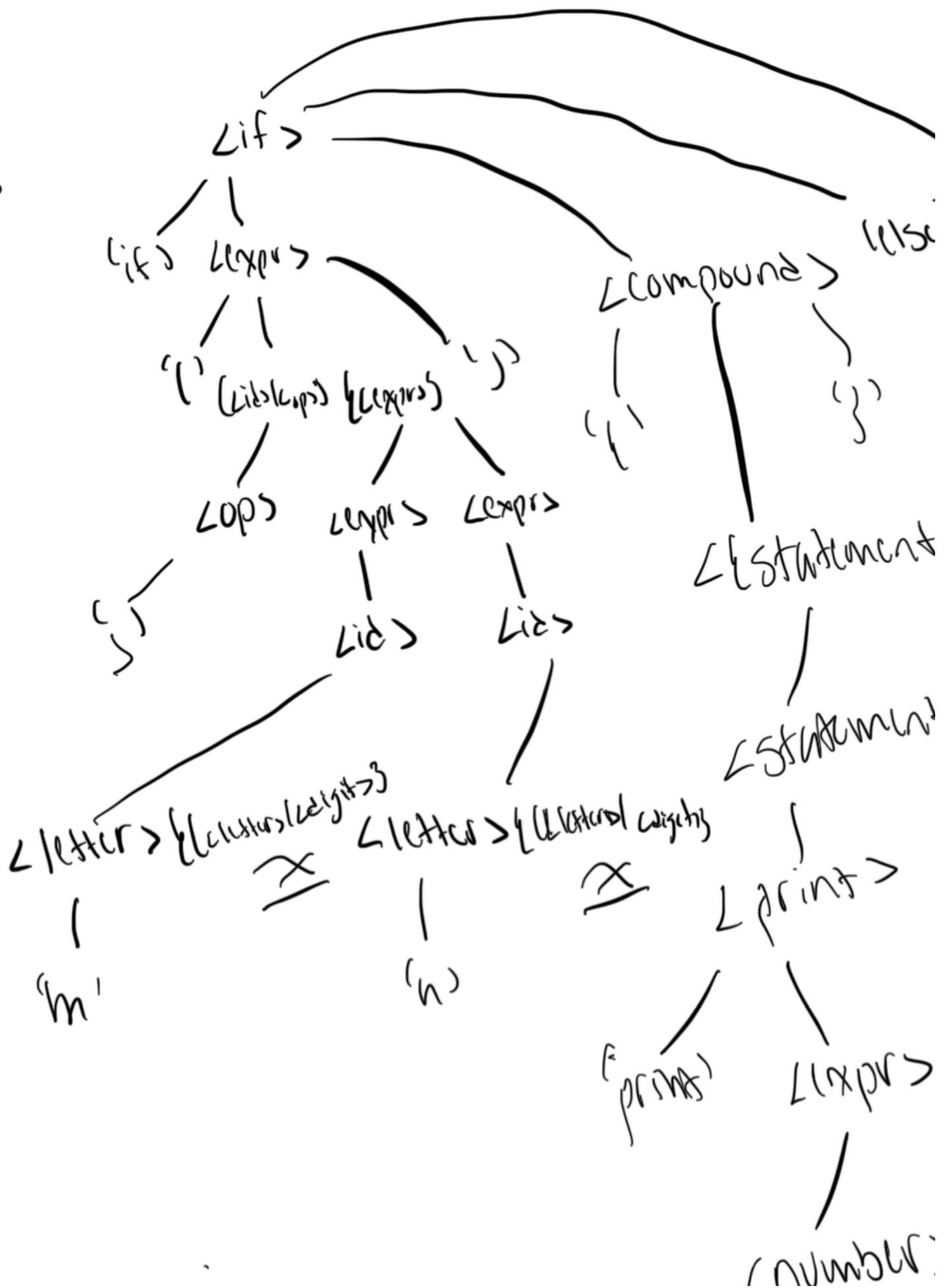
```
1  if true {}
2  else {}
```

is valid, shown by the following parse tree:

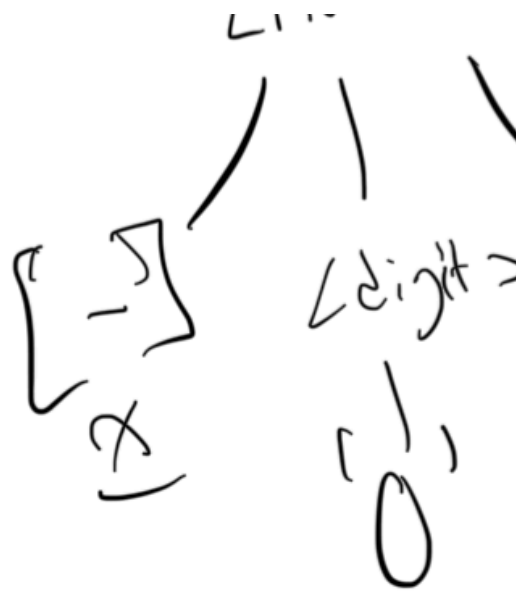


```
1  if (> m n)
2  {
3      print 0
4  }
5  else
6  {
7      print 1
8  }
```

is valid, shown by the parse tree below:







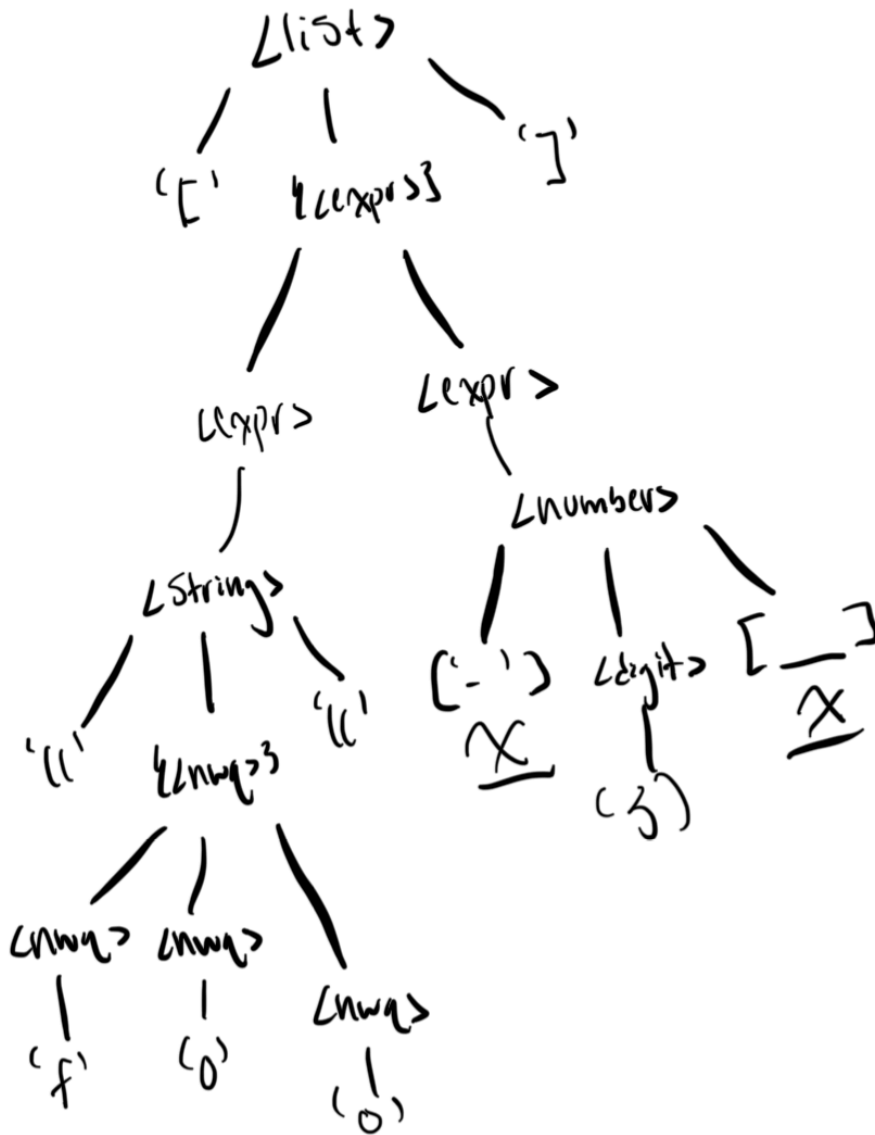
```
1  if (avg >= 80)
2  {
3      if (avg >= 90)
4          print "A"
5      else
6          print "B"
7  }
8  else
9  {
10     print "F"
11 }
```

is invalid because there must be `{}` 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:

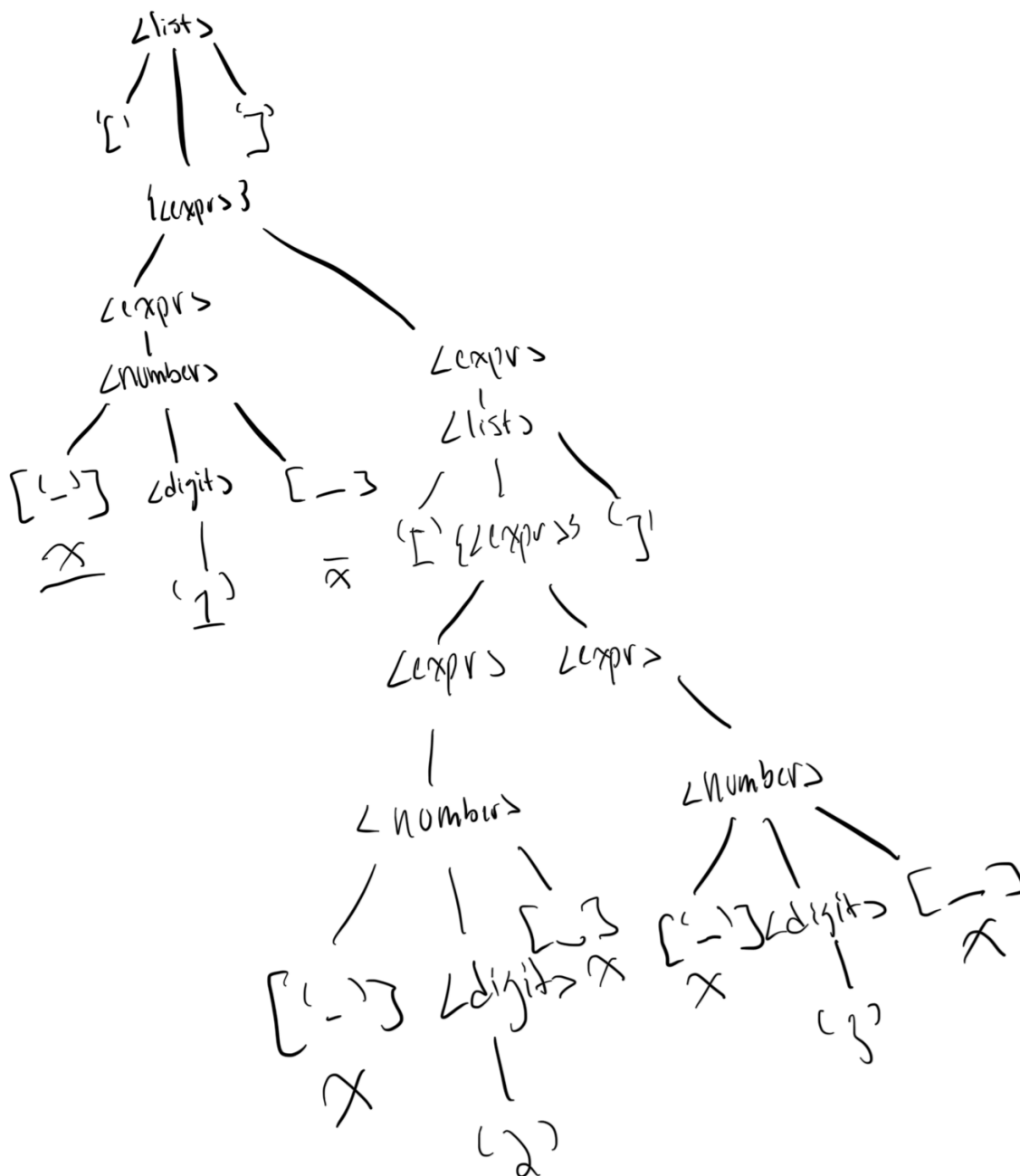


Parse tree for mixed list

## 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 list

## Question 7

Give an example of an assignment statement ( `assign` ) that is as short as possible, in terms of number of tokens.

1    a = 0

## Question 8

Give an example of a while statement ( `while` ) that is as short as possible, in terms of number of tokens.

```
1  while 0{}
```

## Question 9

Give an example of a function declaration ( `func` ) that is as short as possible, in terms of number of tokens.

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