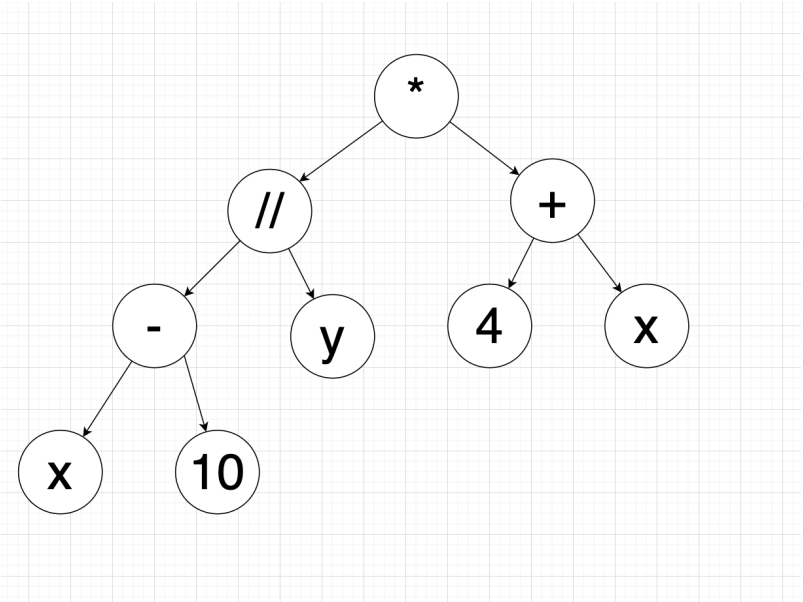
Kyle Connolly, kc5492@rit.edu
Arjun Kozhissery, ak8913@rit.edu

1. ''- // y 2 x''
2. ((y // 2) - x)
3.

    a.



    b.

    ['*', '//', '-', 'x', '10', 'y', '+', '4', 'x']

    c.

    d. (((x - 10) // y) * (4 + x))
    e. (((10-10) // 20) * (4 + 10)) == ((0 // 20) * (14)) == (0 * 14) == 0

```
OPERANDS = ‘+-*//’

def parse(tokens: list) -> Optional[MathNode, LiteralNode, VariableNode]:
    if tokens.empty():
        return None
    token = tokens[0]
    del tokens[0]

    if token in OPERANDS:
        left_child = parse(token)
        right_child = parse(token)
        return MathNode(left_child, right_child, token)

    elif token.isdigit():
        return LiteralNode(int(token)
    elif token.isidentifier():
        return VariableNode(token)
```

```
Tokens = ['+', 'x', 'y’]

Def parse(tokens)
  If tokens is empty:
     Return None

  If tokens[0] in [*, +, /, //, -]:
      Token = tokens[0] // +
    Del tokens[0]
    Left_child = parse(tokens) // LiteralNode(8)
//['+', 'x', 'y’]
    right_Child = parse(tokens)
    Return MathNode(left_Child, right_child, Token)

If token[0].isdigit():
  Num = token[0]
  Del tokens[0]
  Return LiteralNOde(int(num))

If token[0].isidentifier():
  Var = token[0]
  Del token[0]
  Return VariableNode(var)
```