6.101 Recitation 20: Week 12 Symbolic Algebra Wrap-up **4/29/24**

This sheet is yours to keep!

Question 1: Implement the tokenize function from the lab below as a generator. Then, discuss with a neighbor how you might implement parse to work with this new tokenize function.

Examples:

```
tokenize("x") → [‘x’]
tokenize("6.1010") → [‘6.1010’]
tokenize("(x + (-.5 / x))")  → ['(', 'x', '+', '(', '-.5', '/', 'x', ')', ')']

def tokenize(x):
```

Question 2: What if we wanted to be able to make expressions without wrapping every operation in parentheses?

For now, we'll assume that expression now takes in a well-formed string of tokens in one of the following forms:
- single number
- single variable
- one or more symbols surrounded by parentheses, with an operator of equal precedence separating each symbol.

Examples:
```
expression('(x)') -> Var('x')
expression('(1 + 2 + 3)') -> Add(Add(Num(1), Num(2)), Num(3))
expression('(-1 - -2 + 3)') -> Add(Sub(Num(-1), Num(-2)), Num(3))
expression('(-1 - (-2 / x * 5) + 3)') -> Add(Sub(Num(-1), Mul(Div(Num(-2),
Var('x')), Num(3))
```

```
def parse(tokens):  # tokens is a list of strings
```

R20 Participation Credit **Kerberos : ________________@mit.edu**

Hand this sheet in at the end of recitation to get participation credit for today.

Question 3:

What should we do if the expression is not well-formed? Instead of letting Python handle the exception, we are going to try to cleanly raise a custom exception called a SymbolSyntaxError.

a) List different ways an expression could be malformed.

b) Make a plan for how you could detect these various errors (preferably without using try/except).