Case Study: an Expression Tree

```
class ExpressionTree (LinkedBinaryTree):
  def __init(self, token, left=None, right=None):
    super().init()
    self._add_root(token)
    if left is not None:
      self.attach(self.root(), left, right)

 def __str__(self):
   // return str representation of expr
   pieces = []                          // sequence of piecewise strings to compose
   self._parenthesize_recur(self.root(), pieces)
   return ''.join(pieces)

def _parenthesize_recur(self, p, result):
  if self.is_leaf(p)
    result.append(str(p.element()))          // leaf val as str
  else:
   result.append('(')                          // opening parenthesis
   [illegible]
   self._parenthesize_recur(self.left(p), result)   // left subtree
   result.append(p.element())                  // operator
   self._parenthesize_recur(self.right(p), result)  // right subtree
   result.append(')')                          // closing parenthesis
```

Expression Tree Evaluation

```
Algorithm:  if p is a leaf then
                return value at p
            else
                let ∘ be operator stored at p
recursion {     x = evaluate(left(p))
  calls   {     y = evaluate(right(p))
                return x ∘ y
```