Small Project 03

Instructions

[3 points] In either Python or Racket (not both), create a parser that produces a parse tree composed of "nodes", for the following grammar:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Sample BNF Grammar for expressions. Note: this grammar avoids left recursion
;; making it easier to support LL recursive descent parsing.
;;
;; <expr> ::- <term> ADD <expr>
;;              | <term>
;;
;; <term> ::- <factor> MULTIPLY <term>
;;              | <factor>
;;
;; <factor> ::- LPAREN <expr> RPAREN
;;              | NUM
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

[2 points] Write test code to iterate through the parse tree in any order outputing a descriptive string for each node. For example.

Given input: *"(5 + 3) * 8"* without the quotes, the output should be *"5 3 + 8 *"* without the quotes.

Given input: *"5 + 3 * 8"* without the quotes, the output should be *"5 3 8 * +"* without the quotes.