Anthony Weathersby

Antdog_Indahouse@csu.fullerton.edu

CWID:7700

<u>How to run code:</u>

- run python3 parser.py

<u>Overview:</u>

<u>Code Documentation:</u>

Grammar.py:

Holds the required context-free grammar and the parsing table used to trace input strings.

Parser.py:

1. Takes the 3 test inputs.
2. Tokenizes the inputs.
3. Sets the stack to the starting state of 0.
4. Begins a loop to process the tokens.
    a. Parsing Loop:
        i. For each token, it gets the current state from the top of the stack.
        ii. It then finds what action is needed from the parsing table.
        iii. It then increments the step counter and prints out the parsing state (stack contents, remaining input, and the action)
    b. Action Handling:
        i. Shift Actions (S): if action is S, token and new state are pushed onto the stack, then advances the pointer to the next token.
        ii. Reduce Actions (R):
        iii. Accept Action (acc): retrieves the grammar rule, pops elements from the stack based on the number of symbols in the right-hand side of the production rule, pops off twice as many symbols.
        iv. Finds non-terminal symbols that replaces the reduces symbols and retrieves the next state from the parsing table.
        v. Pushes non-terminal and its corresponding state onto the stack.
        vi. If no valid GOTO state exists, returns an error.

Results:

```
Step: 1, Stack: [0], Input: ['(', 'id', '+', 'id', ')', '*', 'id', '$', '$'], Action: S4
Step: 2, Stack: [0, '(', 4], Input: ['id', '+', 'id', ')', '*', 'id', '$', '$'], Action: S5
Step: 3, Stack: [0, '(', 4, 'id', 5], Input: ['+', 'id', ')', '*', 'id', '$', '$'], Action: R6
Step: 4, Stack: [0, '(', 4, 'F', 3], Input: ['+', 'id', ')', '*', 'id', '$', '$'], Action: R4
Step: 5, Stack: [0, '(', 4, 'T', 2], Input: ['+', 'id', ')', '*', 'id', '$', '$'], Action: R2
Step: 6, Stack: [0, '(', 4, 'E', 8], Input: ['+', 'id', ')', '*', 'id', '$', '$'], Action: S6
Step: 7, Stack: [0, '(', 4, 'E', 8, '+', 6], Input: ['id', ')', '*', 'id', '$', '$'], Action: S5
Step: 8, Stack: [0, '(', 4, 'E', 8, '+', 6, 'id', 5], Input: [')', '*', 'id', '$', '$'], Action: R6
Step: 9, Stack: [0, '(', 4, 'E', 8, '+', 6, 'F', 3], Input: [')', '*', 'id', '$', '$'], Action: R4
Step: 10, Stack: [0, '(', 4, 'E', 8, '+', 6, 'T', 9], Input: [')', '*', 'id', '$', '$'], Action: R1
Step: 11, Stack: [0, '(', 4, 'E', 8], Input: [')', '*', 'id', '$', '$'], Action: S11
Step: 12, Stack: [0, '(', 4, 'E', 8, ')', 11], Input: ['*', 'id', '$', '$'], Action: R5
Step: 13, Stack: [0, 'F', 3], Input: ['*', 'id', '$', '$'], Action: R4
Step: 14, Stack: [0, 'T', 2], Input: ['*', 'id', '$', '$'], Action: S7
Step: 15, Stack: [0, 'T', 2, '*', 7], Input: ['id', '$', '$'], Action: S5
Step: 16, Stack: [0, 'T', 2, '*', 7, 'id', 5], Input: ['$', '$'], Action: R6
Step: 17, Stack: [0, 'T', 2, '*', 7, 'F', 10], Input: ['$', '$'], Action: R3
Step: 18, Stack: [0, 'T', 2], Input: ['$', '$'], Action: R2
Step: 19, Stack: [0, 'E', 1], Input: ['$', '$'], Action: acc
String: ( id + id ) * id $:  String is accepted


Step: 1, Stack: [0], Input: ['id', '*', 'id', '$', '$'], Action: S5
Step: 2, Stack: [0, 'id', 5], Input: ['*', 'id', '$', '$'], Action: R6
Step: 3, Stack: [0, 'F', 3], Input: ['*', 'id', '$', '$'], Action: R4
Step: 4, Stack: [0, 'T', 2], Input: ['*', 'id', '$', '$'], Action: S7
Step: 5, Stack: [0, 'T', 2, '*', 7], Input: ['id', '$', '$'], Action: S5
Step: 6, Stack: [0, 'T', 2, '*', 7, 'id', 5], Input: ['$', '$'], Action: R6
Step: 7, Stack: [0, 'T', 2, '*', 7, 'F', 10], Input: ['$', '$'], Action: R3
Step: 8, Stack: [0, 'T', 2], Input: ['$', '$'], Action: R2
```

```
Step: 9, Stack: [0, 'E', 1], Input: ['$', '$'], Action: acc
String: id * id $:  String is accepted

Step: 1, Stack: [0], Input: ['(', 'id', '*', ')', '$', '$'], Action: S4
Step: 2, Stack: [0, '(', 4], Input: ['id', '*', ')', '$', '$'], Action: S5
Step: 3, Stack: [0, '(', 4, 'id', 5], Input: ['*', ')', '$', '$'], Action: R6
Step: 4, Stack: [0, '(', 4, 'F', 3], Input: ['*', ')', '$', '$'], Action: R4
Step: 5, Stack: [0, '(', 4, 'T', 2], Input: ['*', ')', '$', '$'], Action: S7
Step: 6, Stack: [0, '(', 4, 'T', 2, '*', 7], Input: [')', '$', '$'], Action: None
String: ( id * ) $:  String is not accepted
```