Assignment 3 Solution
CMPT432 Spring 2022

Jacob Berendsohn

Jacob.Berendsohn1@Marist.edu

1.   Problem One

Use the following Grammar to derive the given strings:
1 Start â E $
2 E â T plus E
3 | T
4 T â T times F
5 | F
6 F â (E)
7 | num

1.a   Solution

Leftmost grammar derivation. (num plus num times num plus num $)

1. Start

2. E $

3. T plus E $

4. T plus T $

5. T plus T times F $

6. T plus T times E $

7. T plus T times T plus E $

8. T plus T times T plus T $

9. F plus T times T plus T $

10. F plus F times T plus T $

11. F plus F times F plus T $

12. F plus F times F plus F $

13. num plus F times F plus F $

14. num plus num times F plus F $

15. num plus num times num plus F $

16. num plus num times num plus num $

1.b   Solution

Rightmost grammar derivation (num times num plus num times num $)

1. Start

2. E $

3. T $

4. T times F $

5. F times F $

6. E times F $

7. T plus E times F $

8. T times F plus E times F $

9. T times F plus T times F $

10. T times F plus F times F $

11. F times F plus F times F $

12. F times F plus F times num $

13. F times F plus num times num $

14. F times num plus num times num $

15. num times num plus num times num $

1.c   Solution

In terms of left and right precedence, the Non-Terminal <E> is used for the right-associated derivation, in this case T plus E, where E is on the right side of the translation and can be expanded that way. In the case of left-associated derivations we use <T>, in the case T times F the non terminal is on the left side and allows for more translation on the left side.

2.   Problem 2

2.a   Solution

```
parseStart{
    parseValue()
    check('$')
}

parseValue{
    if(nextToken is num){
        check(num)
    } else if(nextToken is 'lparen'){
        check('lparen')
        parseExpr()
        check('rparen')
    } else {
        // Error
    }
}
```

```
18
19 parseExpr{
20     if(nextToken is 'plus'){
21         check('plus')
22         parseValue()
23         parseValue()
24     } else if(nextToken is 'prod'){
25         check('prod')
26         parseValues()
27     } else {
28         // Error
29     }
30 }
31
32 parseValues{
33     if(nextToken is Value){
34         parseValue()
35         parseValues()
36     } else {
37         // Error
38     }
39 }
```

3. Problem 3

CFG : 5 -> S S + | S S * | a
String : aa + a*

3.a Solution

For leftmost derivation:
S -> SS* -> SS+S* -> aS+S* -> aa+S* -> aa+a*

3.b Solution

For rightmost derivation:
S -> SS* -> Sa* -> SS+a* -> Sa+a* -> aa+a*

3.c Solution

Give a parse tree for the provided String:

```
1
2           S
3         / | \
4        S   S   *
5        |     \
6        S       a
7      / | \
8     S   S   +
9     |   |
10    a   a
```