Assignment Two

Logan Hunt

January 27, 2023

1 Question One

1.1 a

There are four productions.

1.2 b

The terminals of this grammar are:

`a b c d`

1.3 c

N_1, N_2, N_3, N_4 are all nonterminals.

1.4 d

The start symbol is N_1.

1.5 e

The symbols N_1, N_2, N_3, N_4 are present in the production heads.

1.6 f

N_2, N_3, N_4, "a", "b", "c", "d" are present in the production bodies.

2 Question Two

2.1 a

There are fifteen productions.

2.2 b

The terminals of this grammar are:

```
0 1 2 3 4 5 6 7 8 9 + -
```

2.3 c

The nonterminals are "list" and "digit".

2.4 d

The start symbol is "list".

2.5 e

"list" and "digit" are present in the production heads.

2.6 f

"list", "digit", 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, − are present in the production bodies.

3 Question Three

```
x -> A (3)
x -> A (3)
xx* -> AA* (2)
(xx*)x+ -> AA+ (1)
((xx*)x+)x+ -> AA+ (1)
```

4 Question Four

4.1 a

Examples:

1. "xy"
2. "xxyy"
3. "xxxyyy"

4. "xxxxyyyy"

This grammar generates a string of x's prepended to a string of y's with the same amount of characters.
The grammar is unambiguous.

4.2 b

Example:

1. "x"
2. "-xx"
3. "+-xx-xx"
4. "-x-xx"

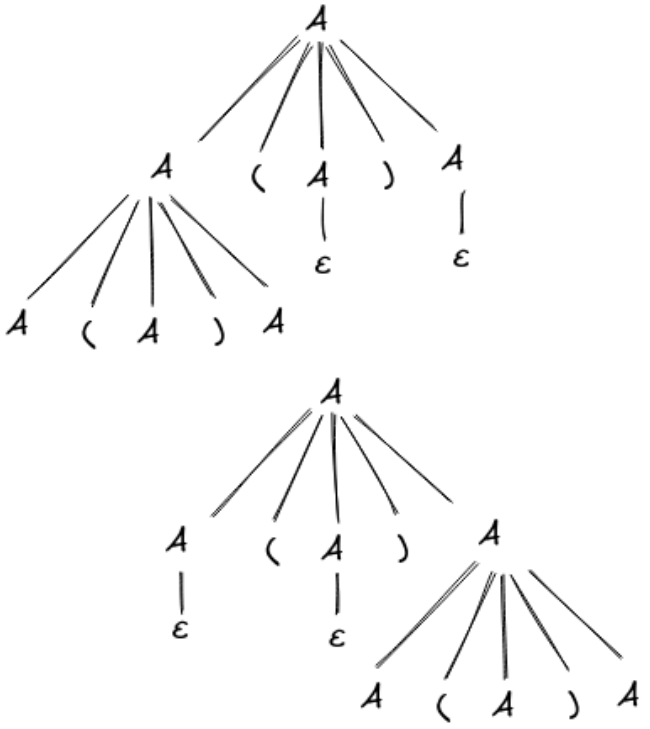
This grammar generates addition and subtraction prefix expressions on "x" or chains of "x".
The grammar is unambiguous.

4.3 c

Example:

1. ""
2. "()"
3. "()()"
4. "(())"

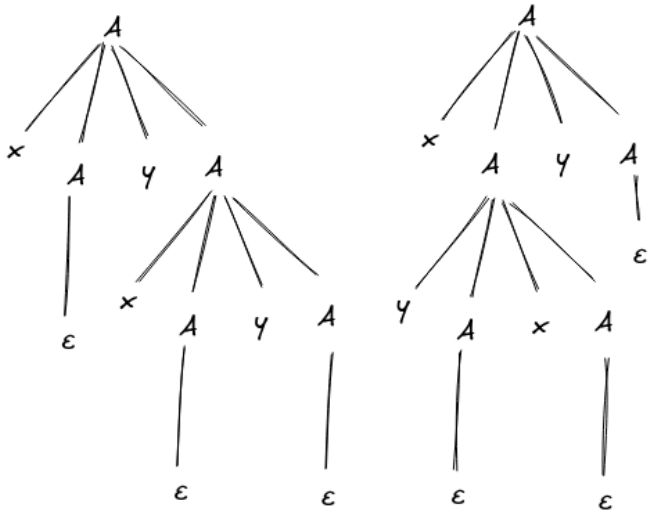
This grammar generates balanced sets of potentially nested parentheses.
The grammar is ambiguous. For example, the string "()()" can be parsed multiple ways:



4.4 d

1. ""
2. "xyyx"
3. "xyxy"
4. "xy"

This grammar generates jumbled combinations of the same number of x's and y's.
The grammar is ambiguous.
For example, the string "xyxy" has two parse trees:



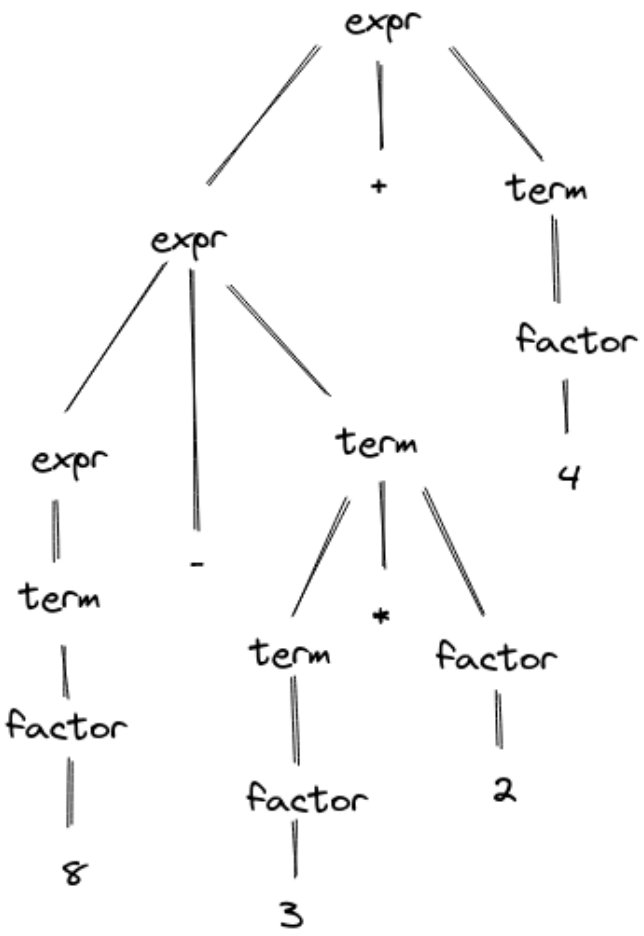
4.5 e

1. "(x)**"
2. "x+x"
3. "x*"
4. "xx"

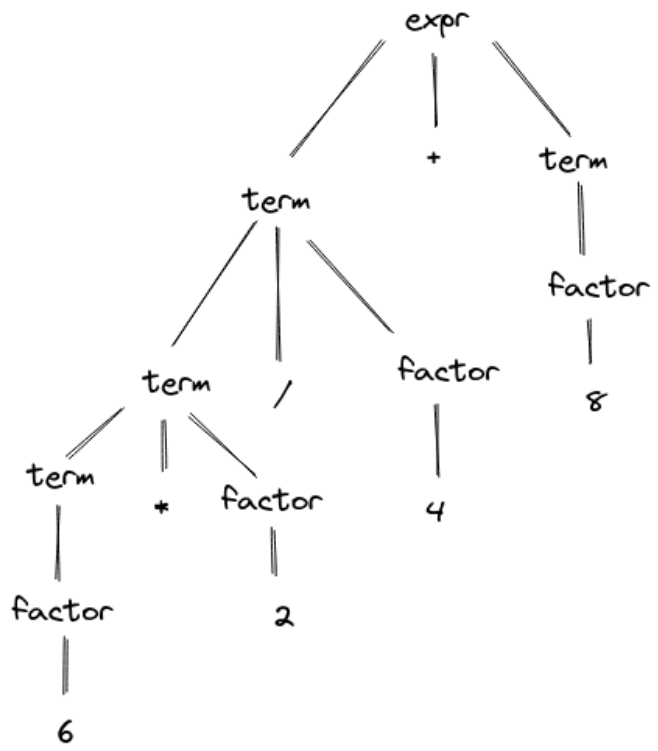
This grammar generates combinations of + / * expressions on x in any nested amount of balanced parentheses.
This grammar is unambiguous.

5 Question Five

5.1 a



5.2 b



6 Question Six

```
expr -> expr expr op
expr -> factor
op -> + | - | / | *
factor -> 0
factor -> 1
factor ... 9
```

7 Question Seven

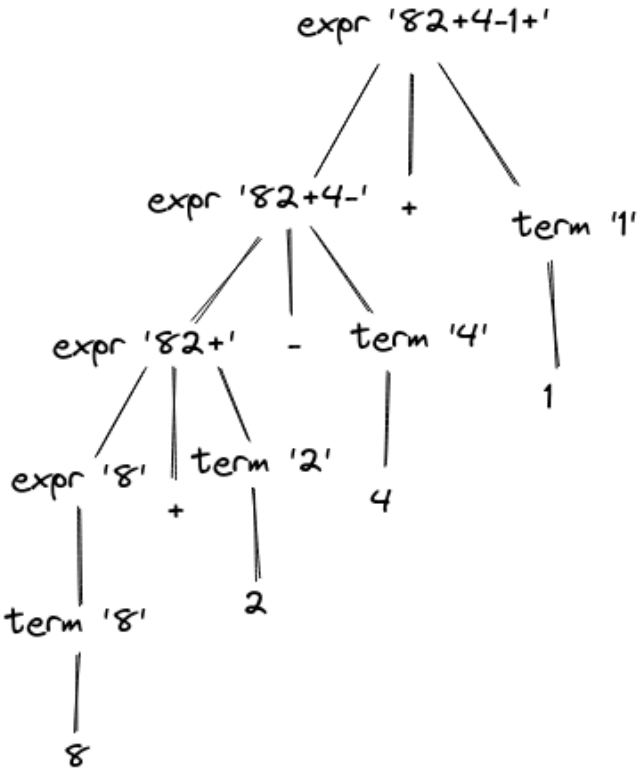
```
sentence -> character
sentence -> character,sentence
character -> a
character -> b
```

```
character -> ...
```

8 Question Eight

```
expr -> expr - term
expr -> expr + term
expr -> expr * term
expr -> expr / term
expr -> term
term -> +num
term -> -num
term -> num
num -> Integer
num -> Ident
```

9 Question Nine

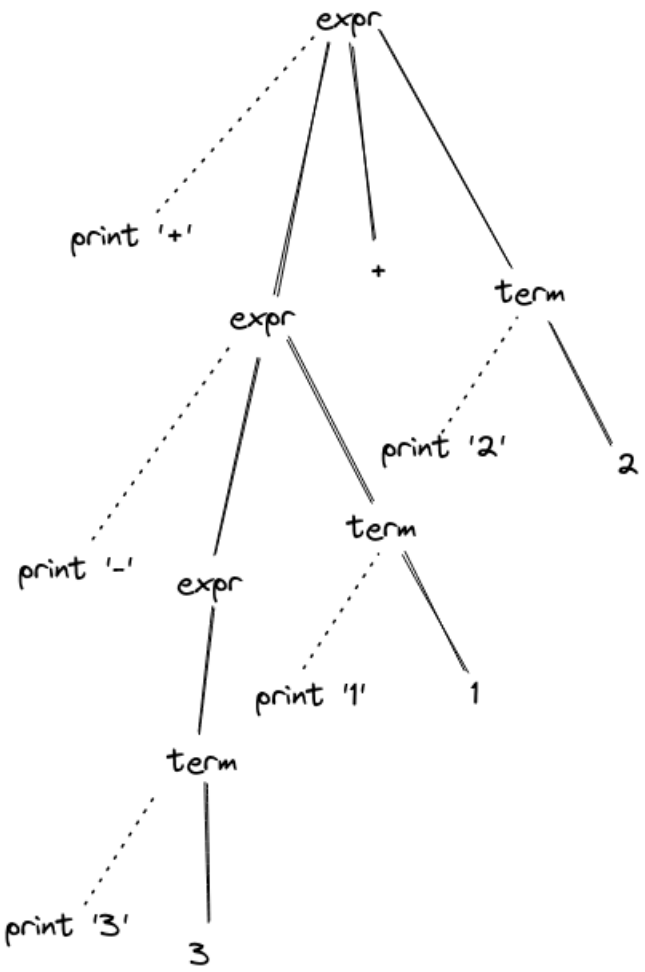


10 Question Ten

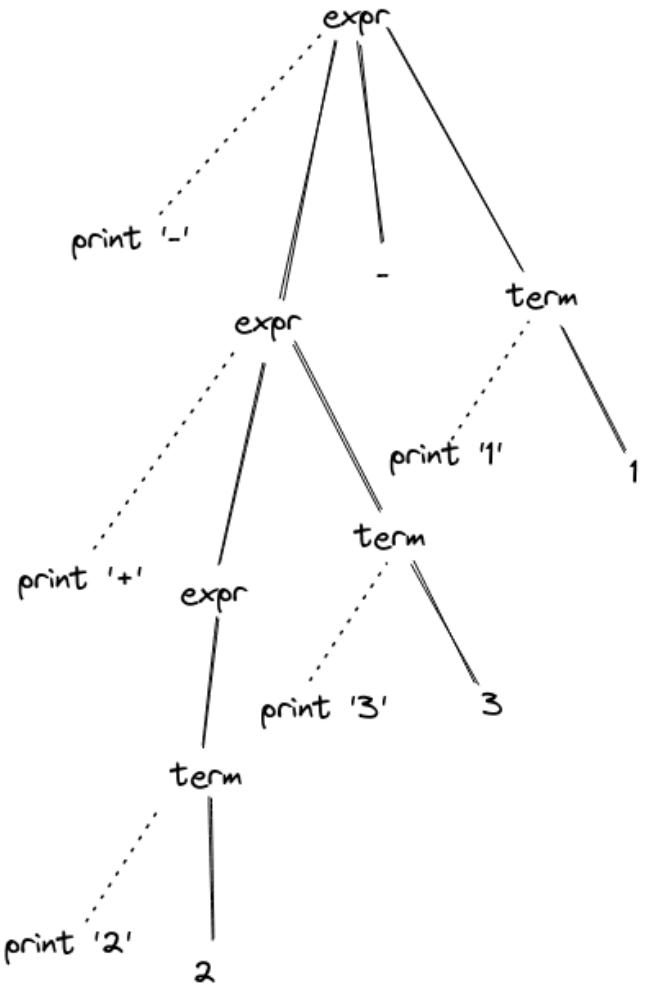
```
expr -> {print '+'} expr + term
expr -> {print '-'} expr - term
expr -> term
term -> 0 {print '0'}
term -> 1 {print '1'}
term -> ... {print ...}
```

11 Question Eleven

11.1 3-1+2



11.2 2+3-1



12 Question Twelve

Predictive parsing is a special kind of recursive descent parsing where the "lookahead unambiguously determines the flow of control".

13 Question Thirteen

None of the productions have the same start symbol (disjoint `FIRST` sets), and it is not left-recursive.

14 Question Fourteen

See `Infix.java`.

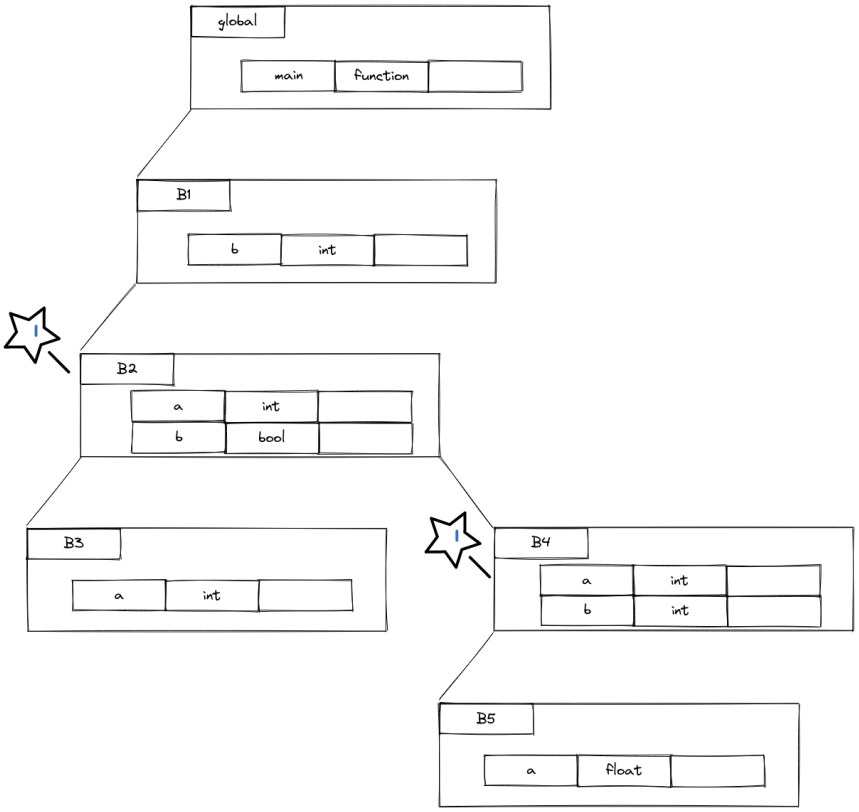
15 Question Fifteen

```
A -> w R
R -> x y z R | \epsilon
```

16 Question Sixteen

```
S -> w R
R -> x y z R
R -> g h R
R -> \epsilon
```

17 Question Seventeen



18 Question Eighteen