Assignment 1 ⇔ CS890D0

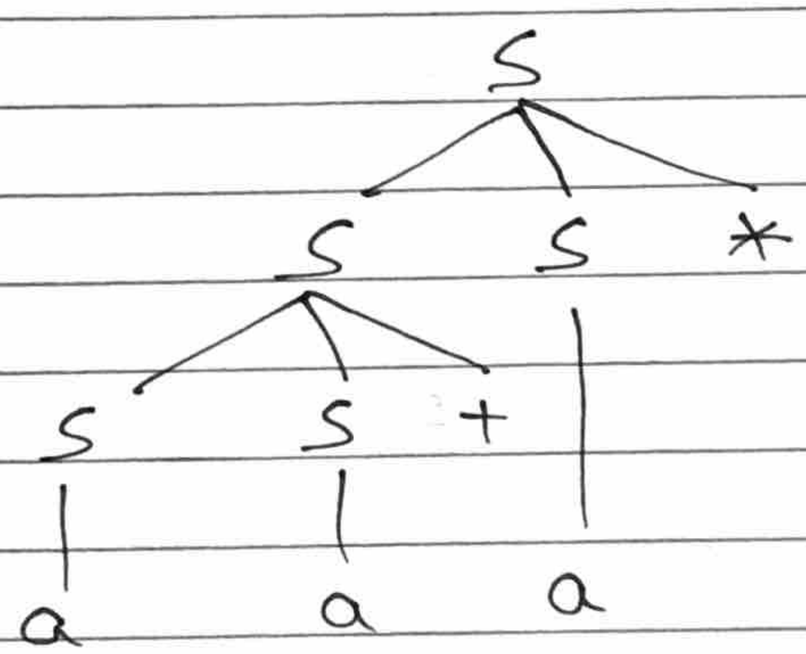
Exercise: 2.2.1 Consider the context free grammar

$$S \rightarrow SS+ \mid SS* \mid a$$

a] Show how string aa+a* can be generated by this grammar.

$$\begin{aligned} S &\rightarrow \underline{S}S* \\ &\rightarrow \underline{S}S+S* \\ &\rightarrow a\underline{S}+S* \\ &\rightarrow aa+\underline{S}* \\ &\rightarrow aa+a* \end{aligned}$$

b] Construct parse tree for this string



c] What language does this grammar generate? Justify your answer.

⇒ This grammar generates postfix expressions consisting of addition and multiplication. Digits are represented by 'a'.

⇒ 'a' is only the ~~non~~terminal symbol in the grammar

So all the 'S' will be replaced by a at the end we will be left of a series or expression consisting of a's and addition and * multiplication signs, in a postfix notation.

Exercise 2.2.2

What language is generated by the following grammars? In each case justify your answer.

a] $S \rightarrow 0S1 \mid 01$

This grammar will have equal no. of 0's and 1's

$$L = \{ 0^n 1^n \mid n >= 1 \}$$

So the smallest string is '01' which has one 0 & one 1.

Some strings generated by the grammar

01, 0011, 000111, ...

$S \rightarrow 0\underline{S}1$
$\rightarrow 00\underline{S}11$
$\rightarrow 000111$

b] $S \rightarrow +SS \mid -SS \mid a$

L = { Prefix expression containing addition and subtraction only with only a as a nonterminal }

So if we want to generate $-+aaa$

$S \rightarrow -\underline{S}\underline{S}$	So this grammar generates string with a's having addition and subtraction signs in prefix notation.
$\rightarrow -+SSS$	
$\rightarrow -+aSS$	
$\rightarrow -+aaS$	
$\rightarrow -+aaa$	

c] $S \rightarrow S(S)S \mid \epsilon$

L = { All properly balanced parantheses which also includes ϵ }

Some of the valid strings :-

(), (()), ()(), ((((()))))....

So for every opening paranthesis there is a closing paranthesis and the string should not start with closing paranthesis.

d] $S \rightarrow aSbS \mid bSaS \mid \epsilon$

$L = \{$ contains equal number of a's and b's any order and contains epsilon $\}$

Some of the strings produced are

ab, ba, abababba, ...

$S \rightarrow aSbS$
$\rightarrow abSaSbS$
$\rightarrow abaSbS$
$\rightarrow ababS$
$\rightarrow abababba$

As we can see we have equal no. of a's and b's in every string produced by grammar.

e] $S \rightarrow a \mid S+S \mid SS \mid S^* \mid (S)$

$L = \{$ Regular expression with terminal a $\}$

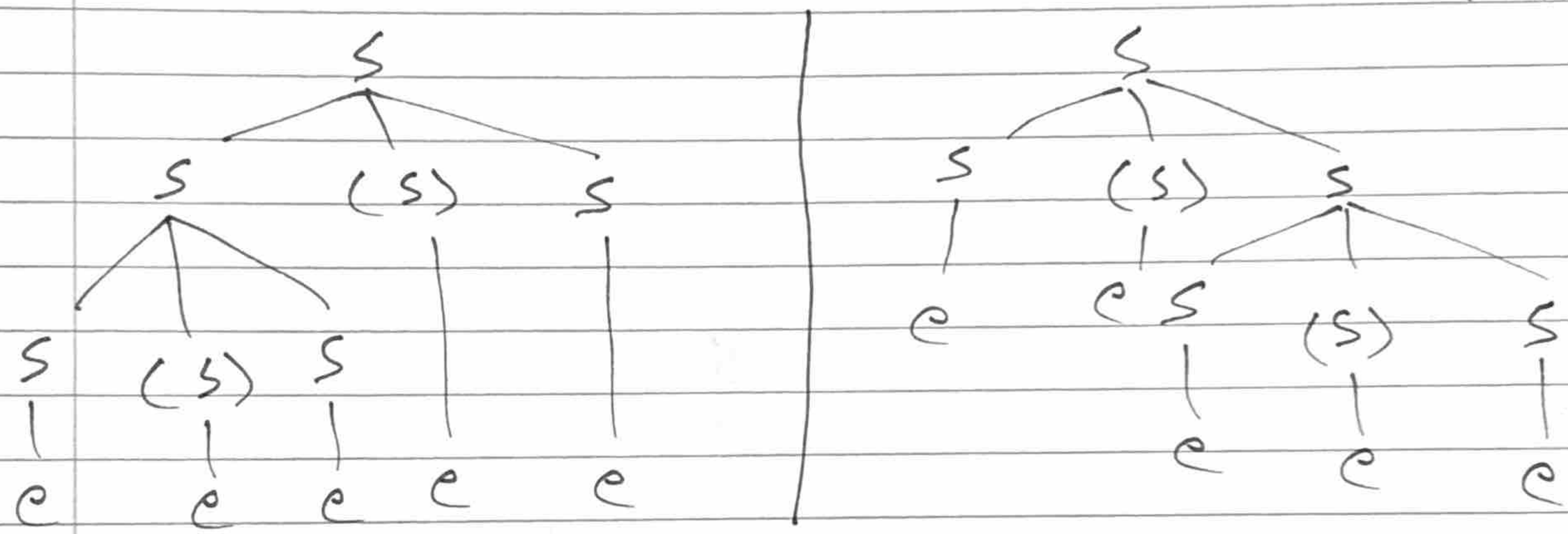
SS : concatenation
S+S : One or more occurance
S^* : zero or more occurance

Exercise 2.2.3

Which of the grammars in exercise 2.2.2 are ambiguous?

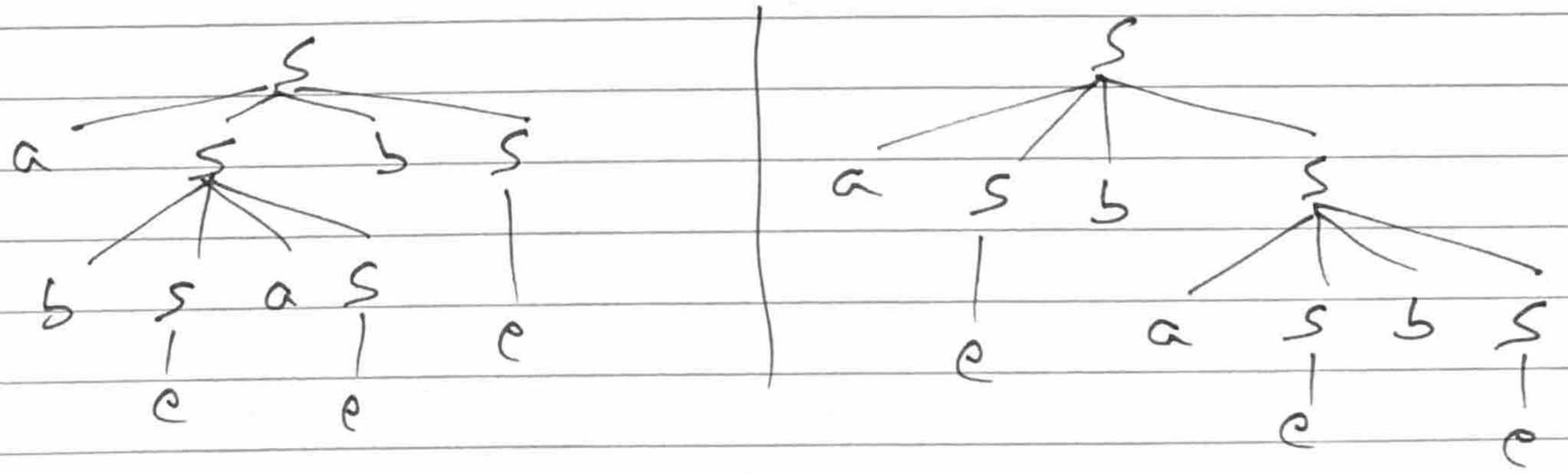
c] $S \to S(S)S \mid \epsilon$ Ambiguous

To generate: ()() we have two parse trees



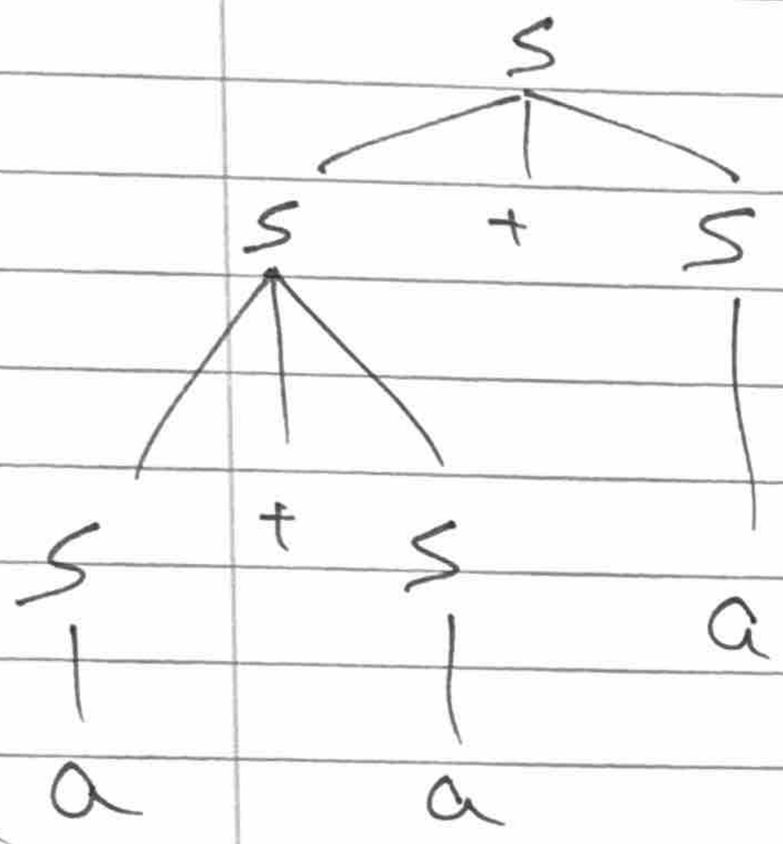
d] $S \to aSbS \mid bSaS \mid \epsilon$ Ambiguous

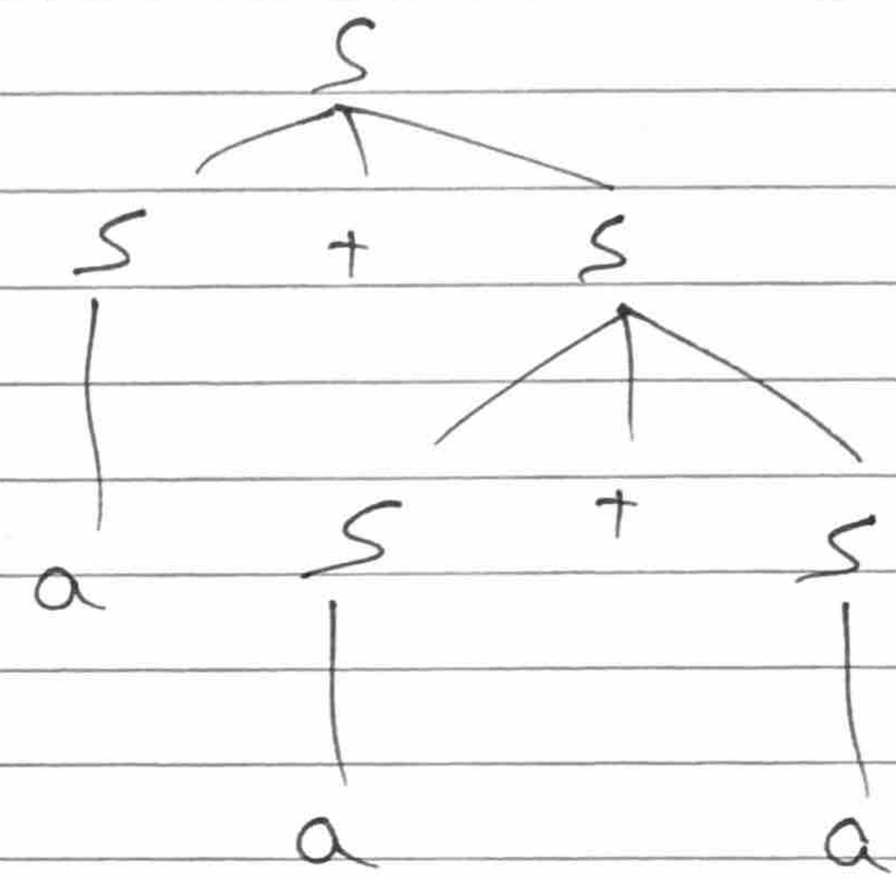
To generate: abab we have two parse trees



e] $S \rightarrow a \mid S+S \mid SS \mid S^* \mid (S)$ → Ambiguos

To generate a+a+a we have two parse trees.





Exercise :- 2.2.4

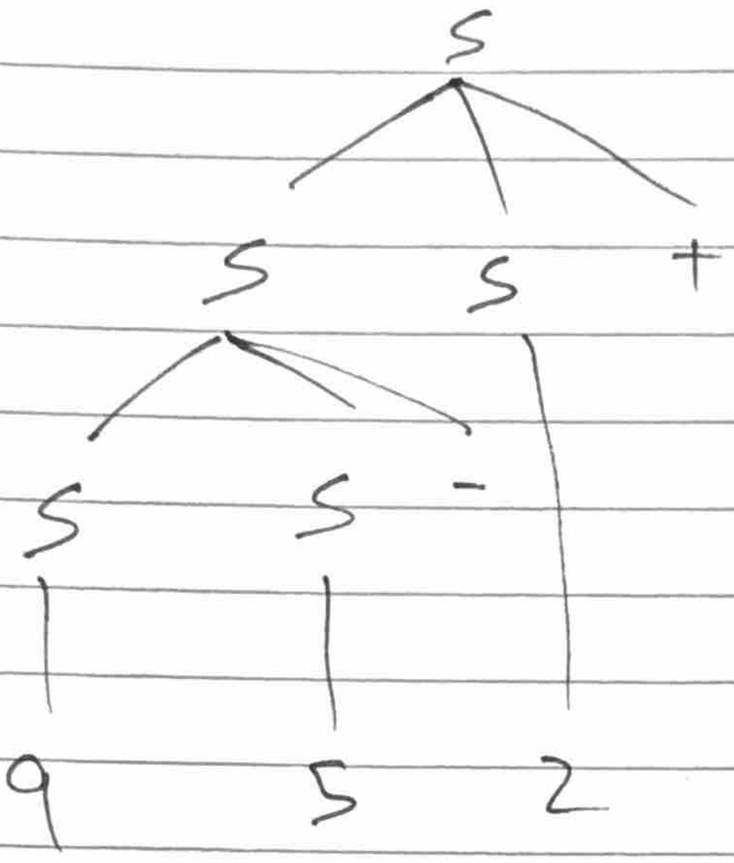
Construct unambiguous context-free grammar for each of following languages. In each case show that your grammar is correct.

a] Arithmetic expressions in postfix notation.

$S \rightarrow SS+ \mid SS- \mid SS* \mid SS/ \mid id$

$id \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

If we want to show 95-2+ then we will have only one parse tree

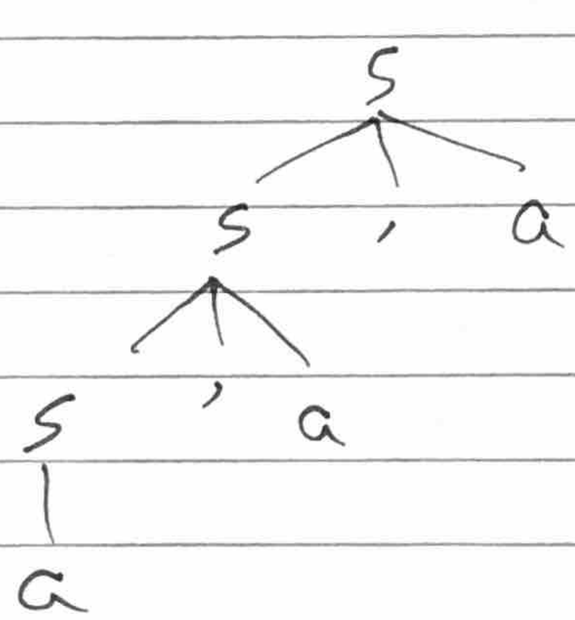


The above parse tree shows that the grammar is correct and unambiguous.

b] Left associative list of identifiers separated by commas.

$$S \longrightarrow S, a \mid a$$

To generate a, a, a we have one parse tree

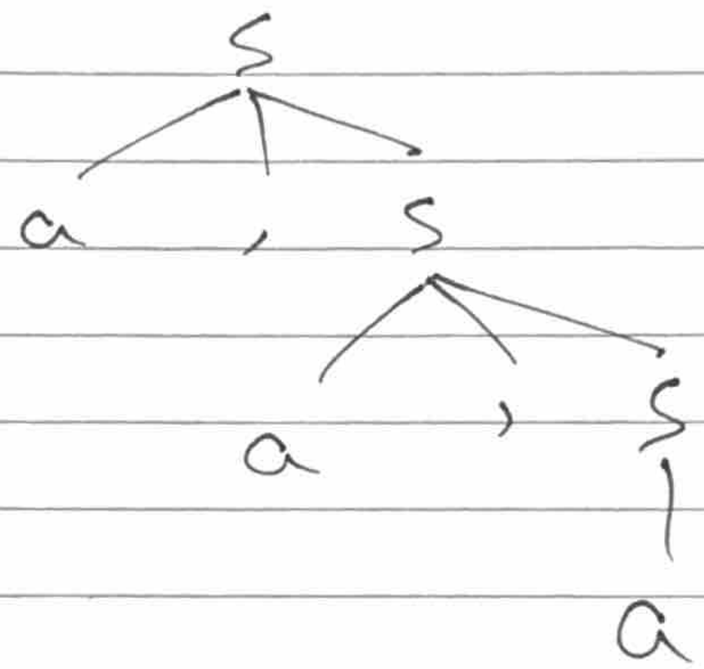


This shows the grammar is correct and unambiguous.

c] Right associative list of identifiers separated by commas.

$$S \rightarrow a, S \mid a$$

to generate a, a, a we have one parse tree



This shows the above grammar as correct and unambiguous.

d] Arithmetic expression of integers and identifiers with the four binary operators +, -, *, /.

$$S \rightarrow S + N \mid S - N \mid N$$

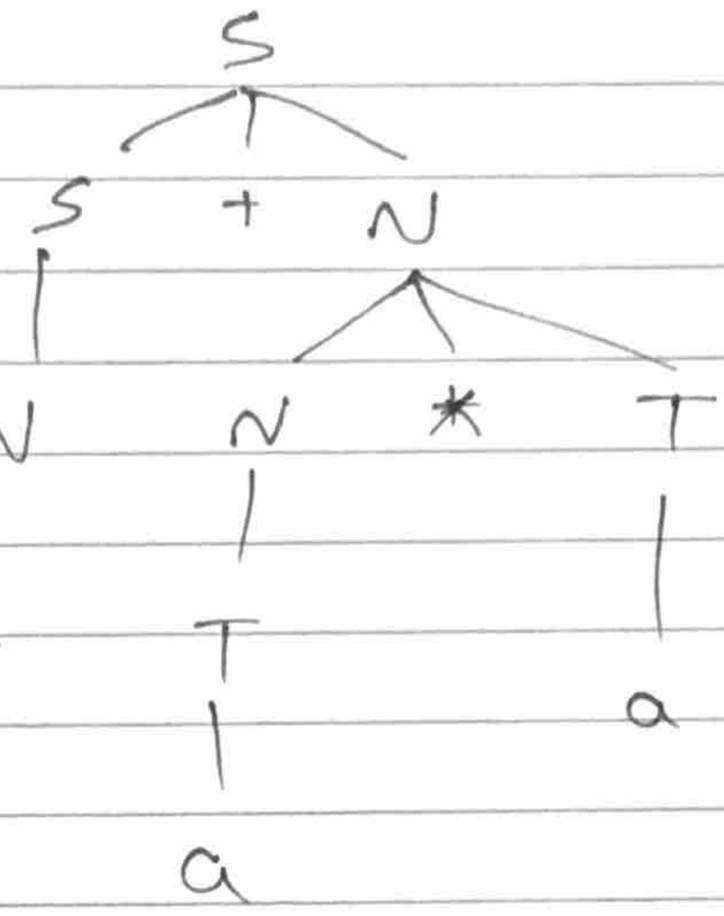
$$N \rightarrow N * T \mid N / T \mid T$$

$$T \rightarrow num \mid a \mid (S)$$

To generate a+a*a, so the multiplication is to performed first.

In parse tree ~~of~~ if any operation ~~an~~ has higher precedence then it should be lower in the tree.

a+a*a



as '*' is lower in the parse tree it will ~~be~~ be executed first.
Also there will be only one parse tree generated for above expression.

e] Add unary plus and minus to the arithmetic operators of (d)

$S \rightarrow S+N \mid S-N \mid N$

$N \rightarrow N*T \mid N/T \mid T$

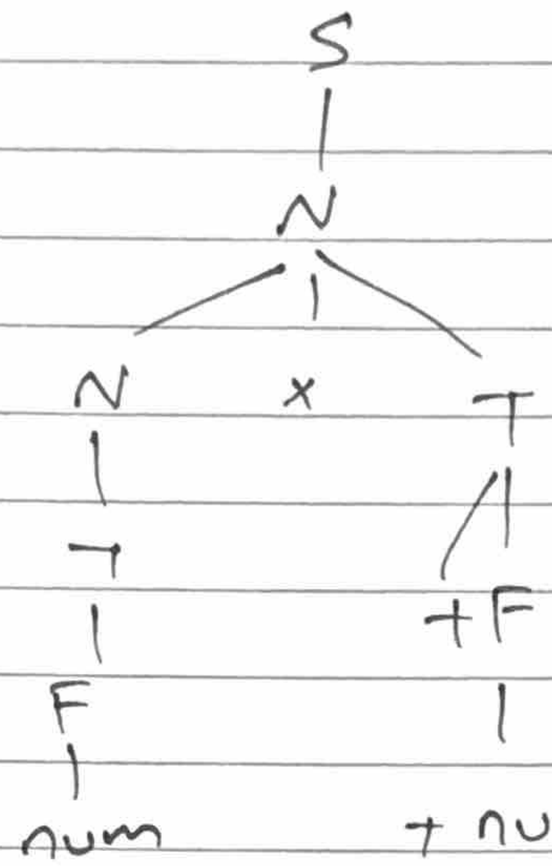
$T \rightarrow +F \mid -F \mid F$

$F \rightarrow id \mid num \mid (S)$

Example:- +num num *num

The unary operator has high precedence as compared to the multiplication. So, it will come at lower level in parse tree then multiplication.

+num num x num



Exercise 2.2.6

Construct a contex-free grammar for romal numerals.

⇒ In decimal the roman value is represented by the following table.

Roman Symbol	I	V	X	L	C	D	M
Desimal value	1	5	10	50	100	500	1000

A number can be generated with the combination of above roman numerals such as

I → 1
II → 2
III → 3
IV → 4
:
VIII → 8
IX → 9

RN → ~~too~~ thousand hundred ten unit

thousand → M | M | MMM | ∈

hundred → sh | CD | D sh | CM

sh → c | cc | ccc | ∈

ten → st | XL | L st | XC

st → x | xx | xxx | ∈

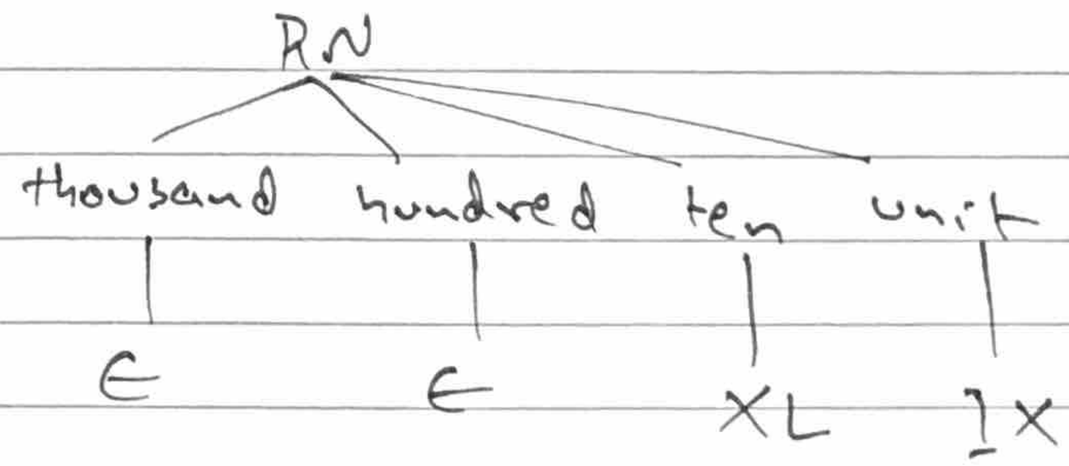
unit → su | IV | V su | IX

su → I | II | III | ∈

Example :- 49

RN → thousand hundred ten unit
→ ∈ hundred ten unit
→ ∈ ten unit
→ XL unit
→ XL IX

Parse Tree



Exercise 2.3.2

Construct a syntax-directed translation scheme that translates arithmetic expression from postfix notation into infix notation. Give annotated parse trees for inputs 95-2*
and 952*-

Following ~~notat~~ grammar is for postfix notation.

$S \rightarrow SF+ \mid SF- \mid F$

$F \rightarrow FT* \mid FT/ \mid T$

$T \rightarrow num \mid (S)$

$num \rightarrow [0 \ldots 9]$

~~Annotated parse tree~~ Translation schemes

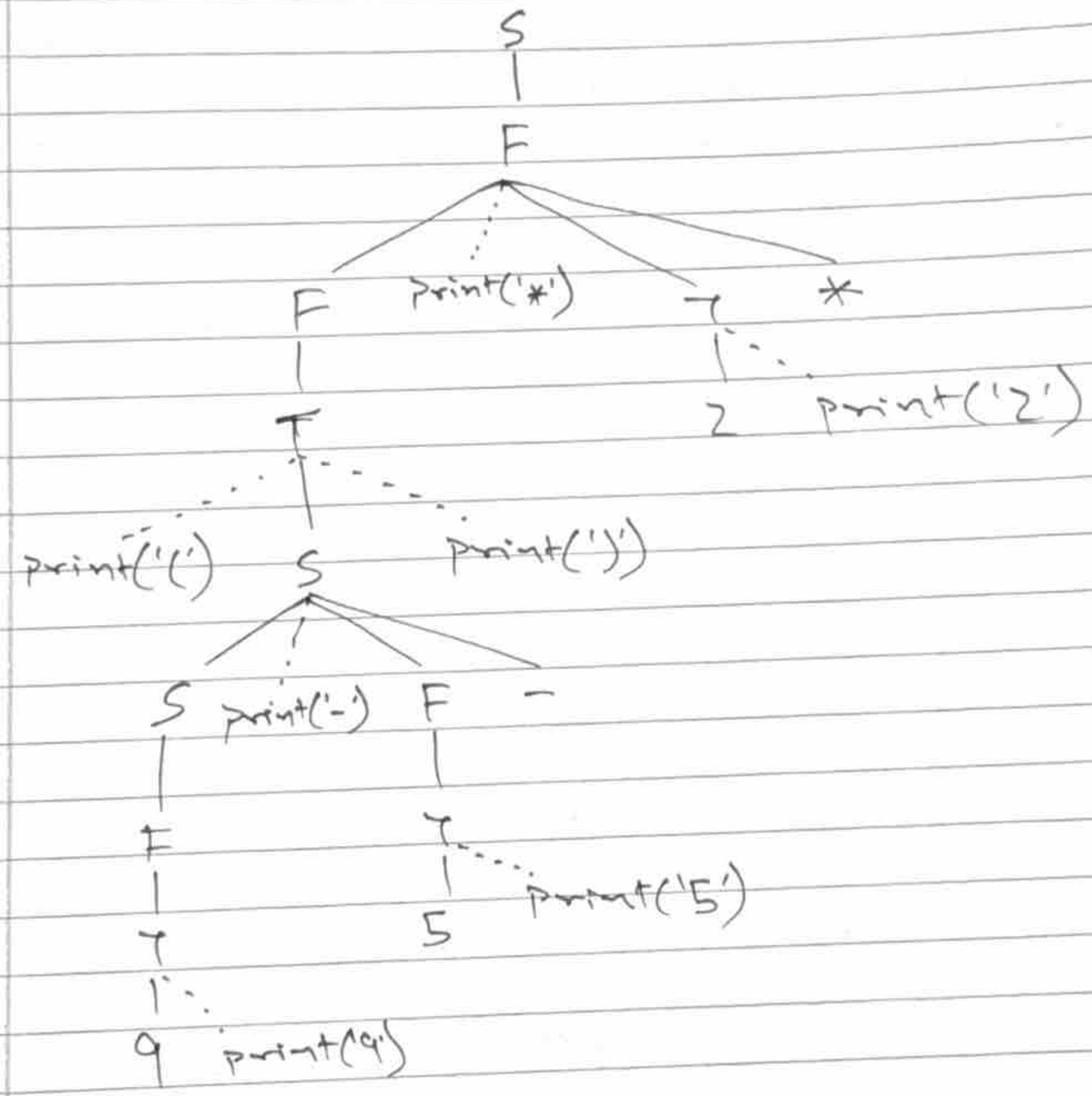
$S \rightarrow S\{print('+')\}F+ \mid S\{print('-')\}F- \mid F$

$F \rightarrow F\{print('*')\}T* \mid F\{print('/')T/ \mid T$

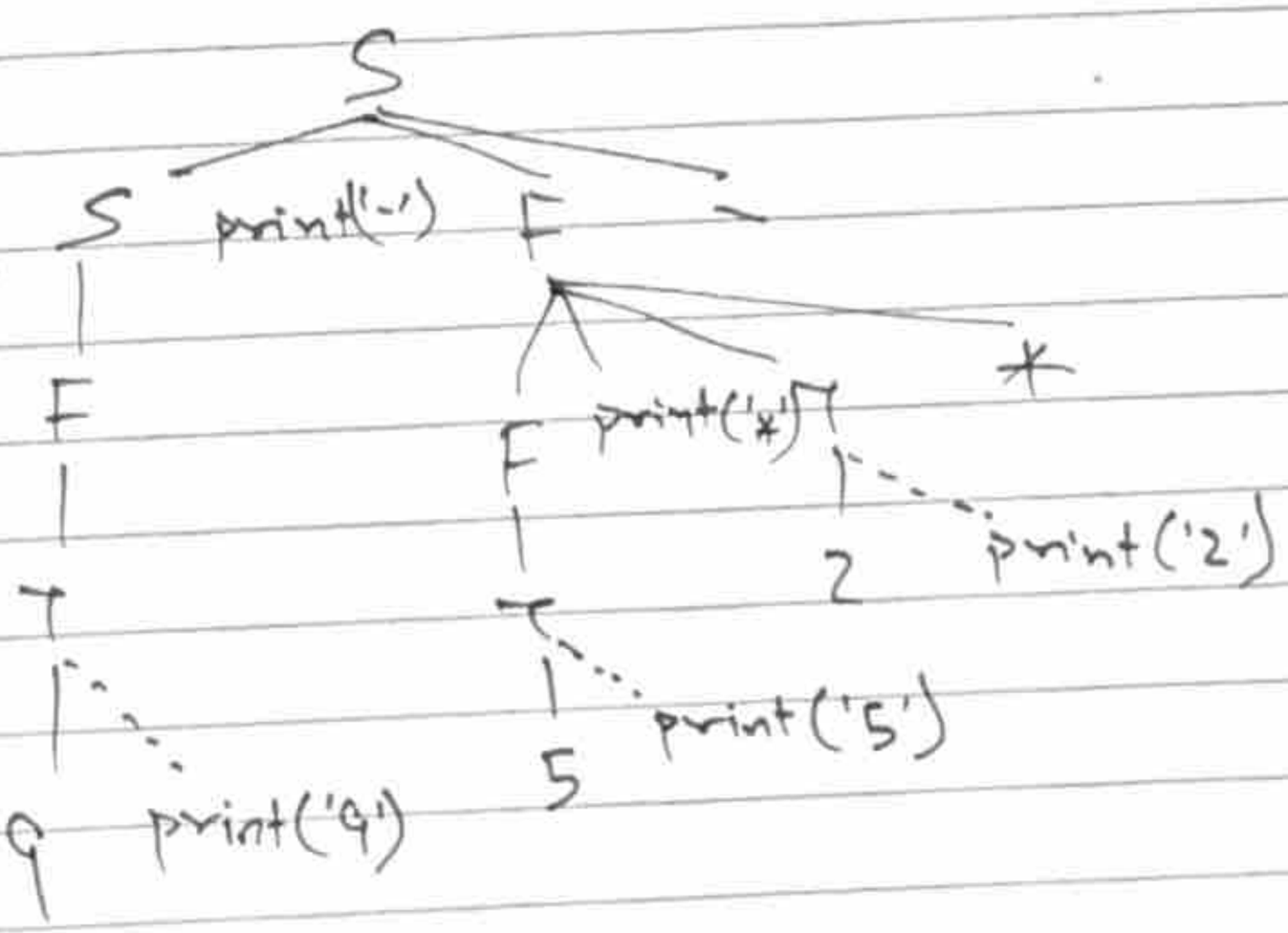
$T \rightarrow num\{print('num.value')\}$

$\mid \{print('(')\}S\{print(')')\}$

⇒ Annotated parse tree for 95-2*



⇒ Annotated parse tree for 952*-



Exercise 2.3.3

Construct a syntax directed translation scheme that translates integers into roman numerals.

num → thousand hundred ten digit
{num.roman= thousand.roman || hundred.roman
|| ten.roman || digit.roman;
print(nu.roman)}

thousand → low{thousand.roman = repeat('M', low.v)}

hundred → low{hundred.roman = repeat('C', low.v)}
| 4{hundred.roman = 'CD'}
| high{hundred.roman = 'D' || repeat('X', high.v-5)}
| 9{hundred.roman = 'CM'}

ten → low{ten.roman = repeat('X', low.v)}
| 4{ten.roman = 'XL'}
| high{ten.roman = 'L' || repeat('X', high.v-5)}
| 9{ten.roman = 'XC'}

digit → low{digit.roman = repeat('I', low.v)}
| 4{digit.roman = 'IV'}
| high{digit.roman = 'V' || repeat('I', high.v-5)}
| 9{digit.roman = IX}

low → 0 {low.v = 0}

| 1 {low.v = 1}

| 2 {low.v = 2}

| 3 {low.v = 3}

high → 5 {high.v = 5}

| 6 {high.v = 6}

| 7 {high.v = 7}

| 8 {high.v = 8}