3.3 Given the grammar

exp → *exp addop term* | *term*

addop → *+* | -

term → *term mulop factor* | *factor*

mulop → ***

factor → *(exp)* | ***number***

write down leftmost derivations, parse trees, and abstract syntax trees for the following ecpressions.

a) 3+4*5-6 b) 3*(4-5+6) c) 3-(4+5*6)

a)

Leftmost derivations:

exp => exp addop term

=> *exp addop term addop term*

=> ***number*** *addop term addop term*

=> ***number*** *+ term addop term*

=> ***number*** *+ term mulop factor addop term*

=> ***number*** *+* ***number*** *mulop factor addop term*

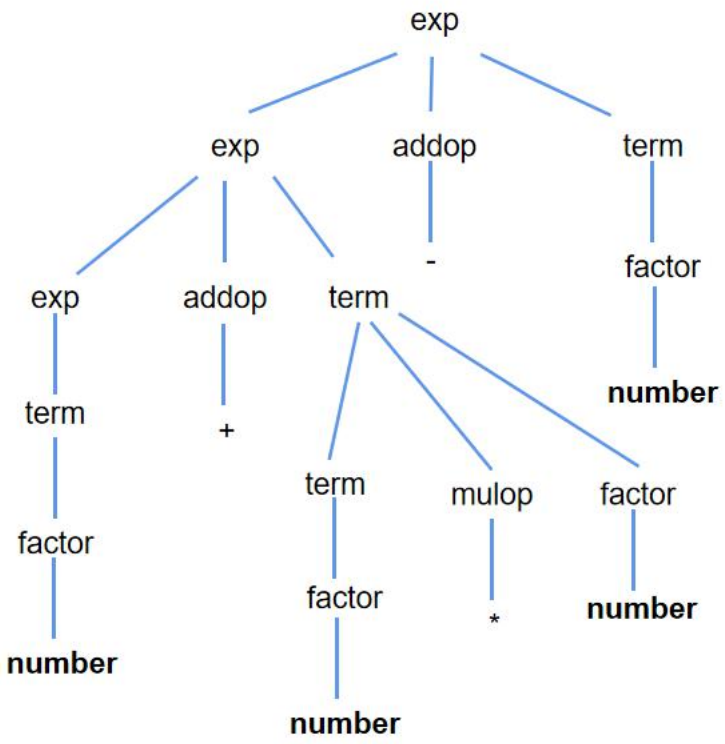
=> ***number*** *+* ***number*** ** factor addop term*

=> ***number*** *+* ***number*** *** ***number*** *addop term*

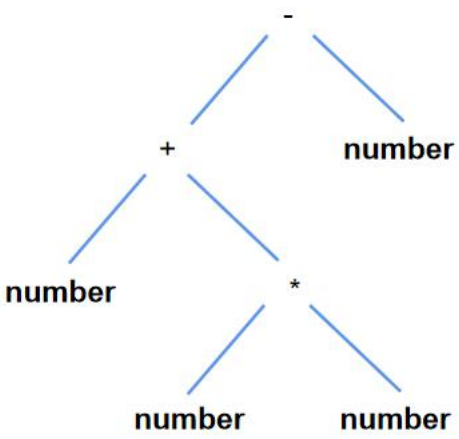
=> ***number*** *+* ***number*** *** ***number*** *+* ***number***

=> ***number*** *+* ***number*** *** ***number*** *+* ***number***

Parse trees:



AST:



b)

Leftmost derivations:

exp => term

=> term mulop factor

=> factor mulop factor

=> ***number*** *mulop factor*

=> ***number*** ** factor*

=> ***number*** * *(exp)*

=> ***number*** * *(exp addop term)*

=> ***number*** * *(exp addop term addop term)*

=> ***number*** * *(term addop term addop term)*

=> ***number*** * *(factor addop term addop term)*

=> ***number*** * *(**number** addop term addop term)*

=> ***number*** * *(**number** + term addop term)*

=> ***number*** * *(**number** + factor addop term)*

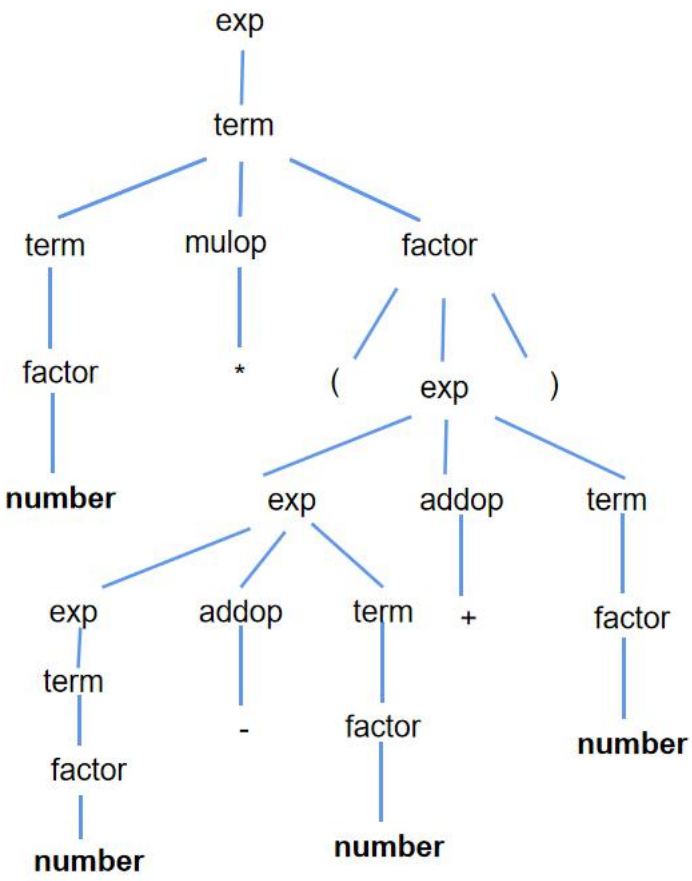
=> ***number*** * *(**number** +**number** addop term)*

=> ***number*** * *(**number** +**number** - term)*

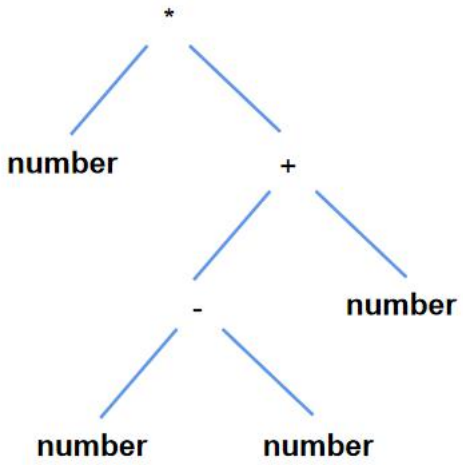
=> ***number*** * *(**number** +**number** - factor)*

=> ***number*** * *(**number** +**number** - **number**)*

Parse trees:



AST:



c)

Leftmost derivations:

exp => exp addop term

=> term *addop term*

=> factor *addop term*

=> ***number*** *addop term*

=> ***number*** *- term*

=> ***number*** *- factor*

=> ***number****- (exp)*

=> ***number****- (exp addop term)*

=> ***number****- (term addop term)*

=> ***number****- (factor addop term)*

=> ***number****- (**number** addop term)*

=> ***number****- (**number** + term)*

=> ***number****- (**number** + term mulop factor)*

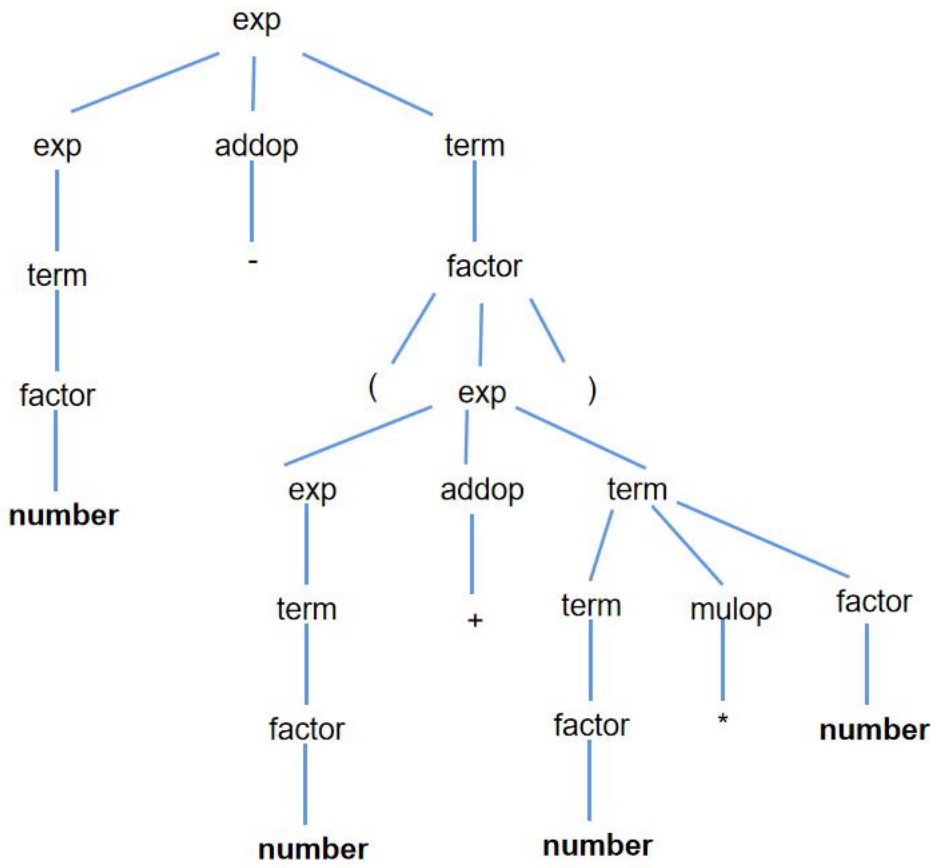
=> ***number****- (**number** + factor mulop factor)*

=> ***number****- (**number** + **number** mulop factor)*

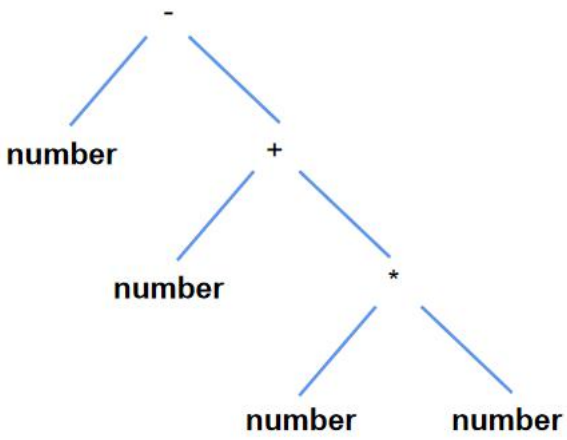
=> ***number****- (**number** + **number** * factor)*

=> ***number****- (**number** + **number** * **number**)*

Parse trees:



AST:



3.4 The following grammar generates all regular expressions over the alphabet of letters (we have used quotes to surround operators, since the vertical bar is an operator as well as a metasymbol) :

rexp → *rexp* " | " *rexp*

| *rexp rexp*
| *rexp “*”*
|*”(“ rexp “)”*
| ***letter***

a) Give a derivation for the regular expression **(ab|b)*** using this grammar
b) Show that this grammar is ambiguous
c) Rewrite this grammar to establish the correct precedence for the operations
d) What associativity does your answer is part(c) give to the binary operators? Why ?

a)

rexp => *rexp “*”*
=> *“(” rexp “)” “*”*
=> *“(” rexp “|” rexp “)” “*”*
=> *“(” rexp rexp “|” rexp “)” “*”*
=> *“(”* ***letter*** *rexp “|” rexp “)” “*”*
=> *“(”* ***letter letter*** *“|” rexp “)” “*”*
=> *“(”* ***letter letter*** *“|”* ***letter*** *“)” “*”*

b)

Another way:
rexp => *rexp “*”*
=> *“(” rexp “)” “*”*
=> *“(” rexp rexp “)” “*”*
=> *“(”* ***letter*** *rexp “)” “*”*
=> *“(”* ***letter*** *rexp “|” rexp “)” “*”*
=> *“(”* ***letter letter*** *“|” rexp “)” “*”*
=> *“(”* ***letter letter*** *“|”* ***letter*** *“)” “*”*

c)

rexp → *rexp* “|” *rnon* | *rnon*
rnon → *rnon rterm* | *rterm*
rterm → *rterm “*”* | *item*
item → **letter** | *“(” rexp “)”*

d)

Left associativity.