1. Section 1

Give an unambiguous grammar to the language of arithmetic expressions consisting of variable identifiers, integer constants, float-point constants, addition (+), subtraction (-), multiplication (*), division (/), and exponentiation (#) operators as well as parentheses. Adopt usual convention for operator precedence and associativity. Note that (#) has the highest precedence and associates to the right. Assume that id, integer and float are tokens.

Order of Operations

1. Parentheses
2. Power
3. Multiplication or division (Left - Right)
4. Addition or Subtration (Left - Right)

$$
\begin{aligned}
&< addsubexp > ::=< addsubexp > + < muldivexp > \mid < addsubexp > - < muldivexp > \mid < muldivexp > \\
&< muldivexp > ::=< muldivexp > * < expexp > \mid < muldivexp > / < expexp > \mid < expexp > \\
&\quad < expexp > ::=< rootexp > \# < expexp > \mid < rootexp > \\
&\quad < rootexp > ::= (< addsubexp >) \mid id \mid integer \mid float
\end{aligned}
\tag{1.1}
$$

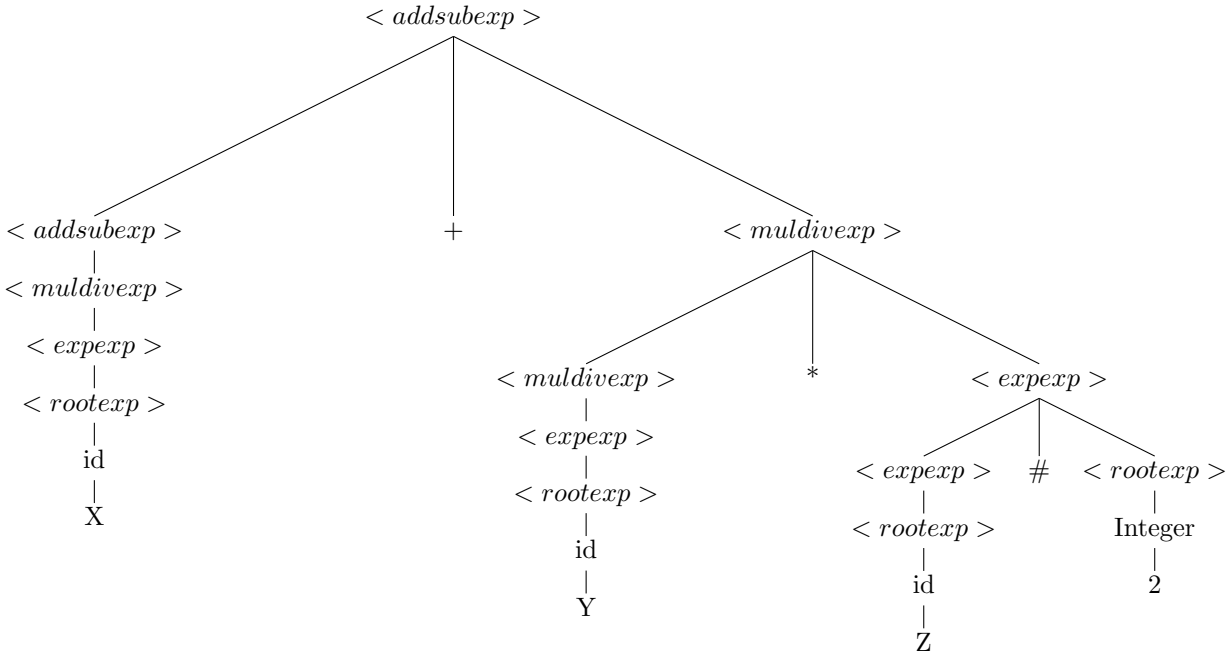
2. Section 2

Using the above grammar, draw a parse tree and an abstract syntax tree for each of the following expressions where English letters are variable identifiers.

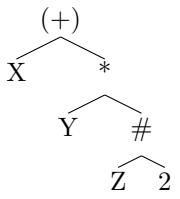
2.1 a

$$X + Y * Z\#2$$

Parse Tree



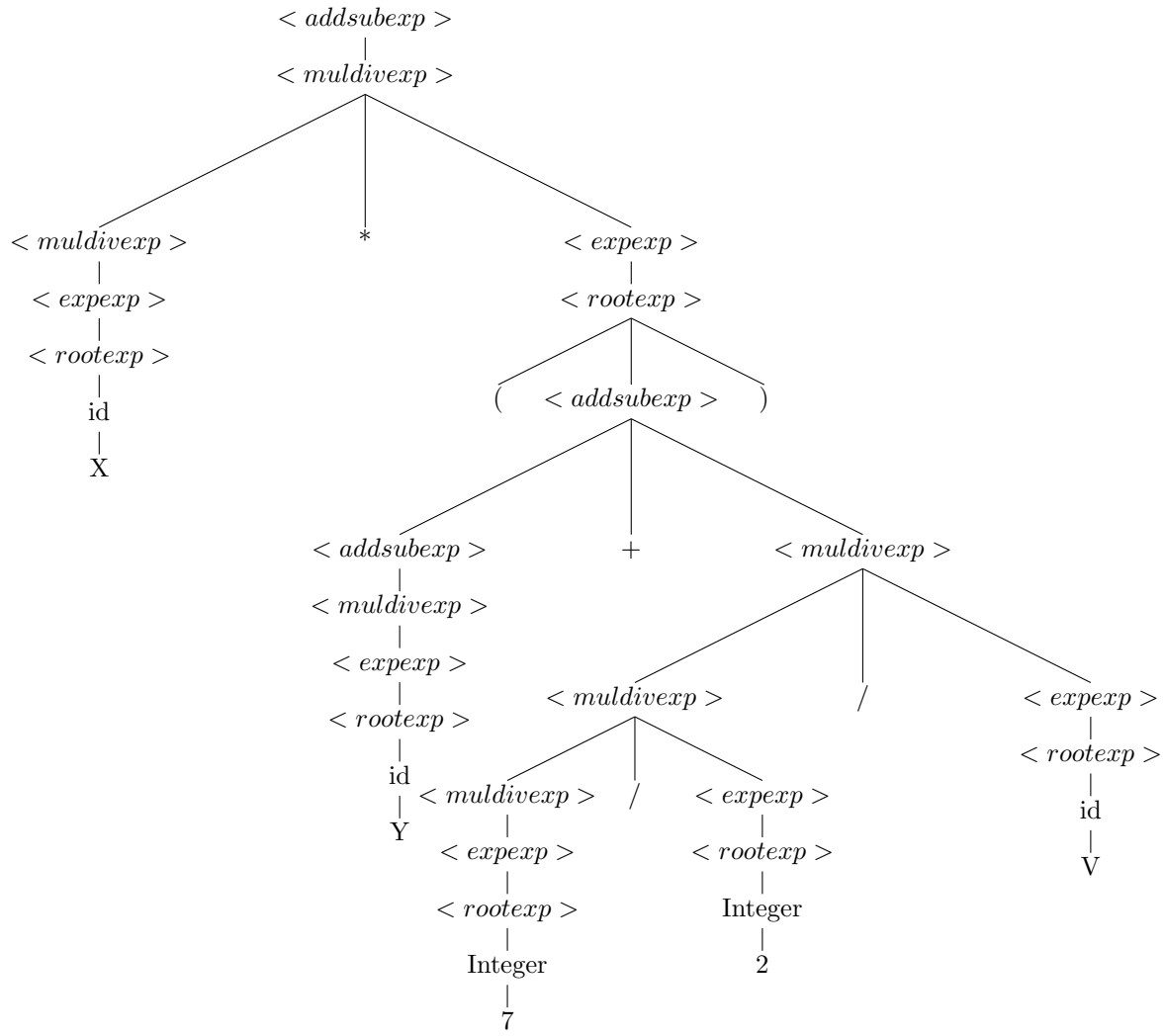
Abstract Syntax



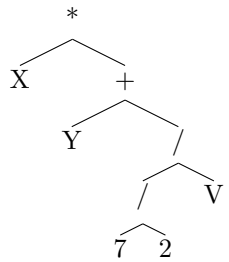
2.2 b

$$X * (Y + 7/2/V)$$

Parse Tree



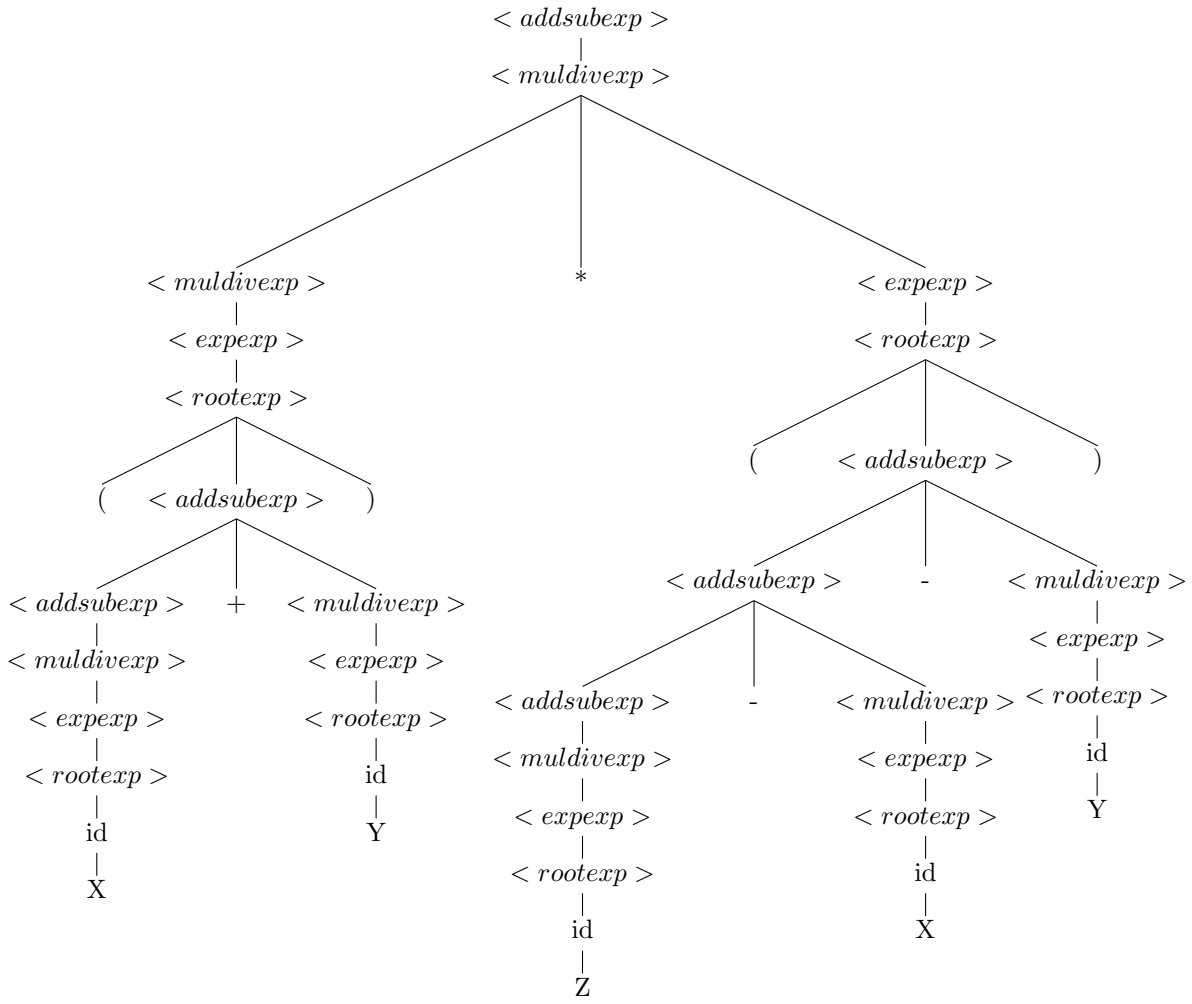
Abstract Syntax



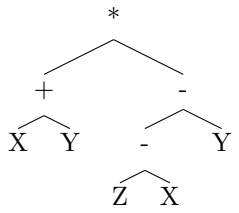
2.3 c

$$(X + Y) * (Z - X - Y)$$

Parse Tree



Abstract Syntax



3. Section 3

Extend the grammar in item 1 so that the language contains Boolean expressions formed of comparison operators == and >= applied to arithmetic expressions and Boolean operators &&, || and ! (not) applied to Boolean expressions. The operator ! (logical negation) is prefix and is of higher precedence than && (logical conjunction) that has higher precedence than || (logical disjunction).

Order of Operations

1. Parentheses
2. Power
3. Multiplication or division (Left - Right)
4. Addition or Subtration (Left - Right)
5. == or >=
6. !
7. &&
8. ||

$$
\begin{aligned}
&< logconj > ::=< logconj > || < logdisj > | < logdisj > | < addsubexp > \\
&< logdisj > ::=< logdisj > \&\& < logneg > | < logneg > \\
&< logneg > ::=! < compopt > | < compopt > \\
&< compopt > ::=< addsubexp >==< addsubexp > | < addsubexp >>=< addsubexp > \\
&< addsubexp > ::=< addsubexp > + < muldivexp > | < addsubexp > - < muldivexp > | < muldivexp > \\
&< muldivexp > ::=< muldivexp > * < expexp > | < muldivexp > / < expexp > | < expexp > \\
&< expexp > ::=< rootexp > \# < expexp > | < rootexp > \\
&< rootexp > ::= (< logconj >)|id|integer|float
\end{aligned}
\tag{3.1}
$$