I understand you want the LL(1) parsing steps (stack, input, actions) for the expression y = a + b * c; with the input formatted as y = a + b (*) c; \$, where (*) likely indicates a multiplication operation, possibly with parentheses in the intended expression, but based on your previous examples, I'll assume you mean y = a + b * c; with \$ as the end-of-input marker. The (*) seems to be a typo or emphasis for the * operator, so I'll parse y = a + b * c; using the LL(1) grammar from the previous response. If you meant something else (e.g., actual parentheses like y = a + b * (c);), please clarify.

I'll provide only the stack, input, and actions for LL(1) parsing, as requested, using the same grammar as before for consistency. The grammar is:

```
text

\begin{array}{c}
\times & \text{Collapse} & \Rightarrow \text{Wrap} & \bigcirc \text{Copy} \\
\end{array}

\begin{array}{c}
S \rightarrow \text{id} = E; \\
E \rightarrow T E' \\
E' \rightarrow + T E' \mid E \\
T \rightarrow F T' \\
T' \rightarrow * F T' \mid E \\
F \rightarrow \text{id} \mid (F)
\end{array}
```

- * Terminals: id (identifier, e.g., y, a, b, c), =, +, \star , (,), ;, \$ (end of input).
- *Non-terminals: S (statement), E (expression), E' (expression tail), T (term), T' (term tail), F (factor).

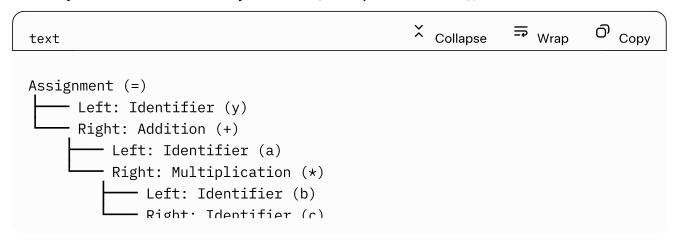
Parsing y = a + b * c;

- Input tokens: y, =, a, +, b, *, c, ;, \$
- * Parsing steps: Start with stack S \$.

Step	Stack	Input	Action O
1	S \$	y = a + b * c;\$	S → id = E ;
2	id = E ; \$	y = a + b * c;\$	Match id (y)
3	= E;\$	= a + b * c;\$	Match =
4	E;\$	a+b*c;\$	E → T E'
5	T E';\$	a + b * c;\$	T → F T'
6	F T' E' ; \$	a+b*c;\$	F → id
7	id T' E' ; \$	a + b * c;\$	Match id (a)
8	T' E' ; \$	+ b * c;\$	T' → ε
9	E';\$	+ b * c;\$	E' → + T E'
10	+ T E' ; \$	+ b * c;\$	Match +
11	T E';\$	b*c;\$	T → F T'
12	F T' E' ; \$	b*c;\$	F → id
13	id T' E' ; \$	b*c;\$	Match id (b)
14	T' E' ; \$	*c;\$	T' → * F T'
15	* F T' E' ; \$	*c;\$	Match *
16	F T' E' ; \$	c;\$	F → id
17	id T' E' ; \$	c;\$	Match id (c)
18	T' E' ; \$;\$	T' → ε
19	E';\$;\$	E' → ε
20	;\$;\$	Match ;
21	\$	\$	Accept

Notes

- Input Format: I assumed (*) in y = a + b (*) c; \$ emphasizes the * operator, so I parsed y = a + b * c; If you meant y = a + b * (c); the grammar supports it, but the parsing steps would differ slightly due to the parentheses. Let me know if you want that version.
- * Grammar: The LL(1) grammar handles + and * with proper precedence (* binds tighter than +). The ε productions allow the parser to terminate expression tails when no further operators are present.
- * AST: If you want the abstract syntax tree (as requested earlier), it's:



* Clarification: If you want another example with different operators, parentheses, or the full original expression x = (a + b) * (c - d) / 2.5 + e % f;, I can provide it, but it requires a more complex grammar to handle $\frac{1}{2}$, $\frac{1}{2}$, and $\frac{1}{2}$. Let me know your preference!

If you need more examples or a specific variation (e.g., with parentheses or other operators), please specify!