CS3342 – Assignment 2 due Feb. 17, 2024

2-day no-penalty extension until: Feb. 19, 11:59pm

1. (30pt) Consider a language where assignments can appear in the same context as expressions; the value of a = b = c equals the value of c. The following grammar, *G*, generates such expressions that includes assignments in addition to additions and multiplications:
   1. *program −→ exp* $$
   2. *exp −→* id = *exp*
   3. *exp −→ term term tail*
   4. *term tail −→* + *term term tail*
   5. *term tail −→ ε*
   6. *term −→ factor factor tail*
   7. *factor tail −→* \* *factor factor tail*
   8. *factor tail −→ ε*
   9. *factor −→* ( *exp* )
   10. *factor −→* id
2. (3pt) Show a parse tree for the string: id = id \* (id = id + id \* id)$$.
3. (10pt) For each production *A −→ α*, compute first(*α*) and follow(*A*) using the algorithm below;

first(*α*) is computed by string FIRST(*α*). For each token added, indicate the pair (*step, prod*) used

to add it, where 0 *≤ step ≤* 3 is the step in the algorithm (marked as , , , below) and

0

1

2

3

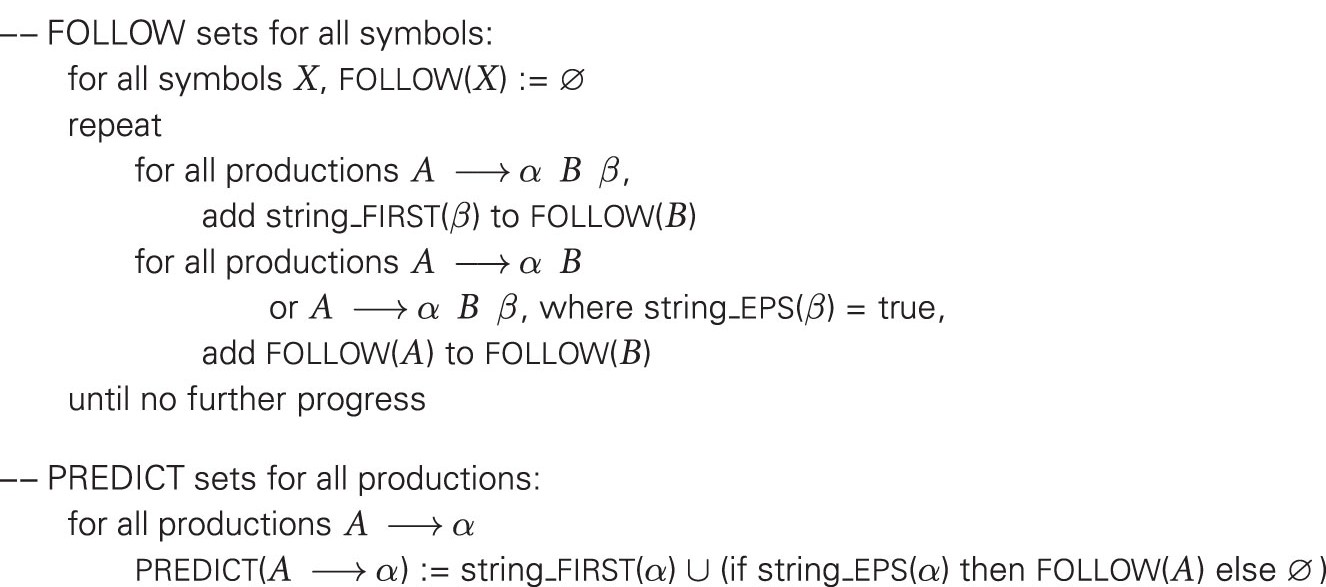
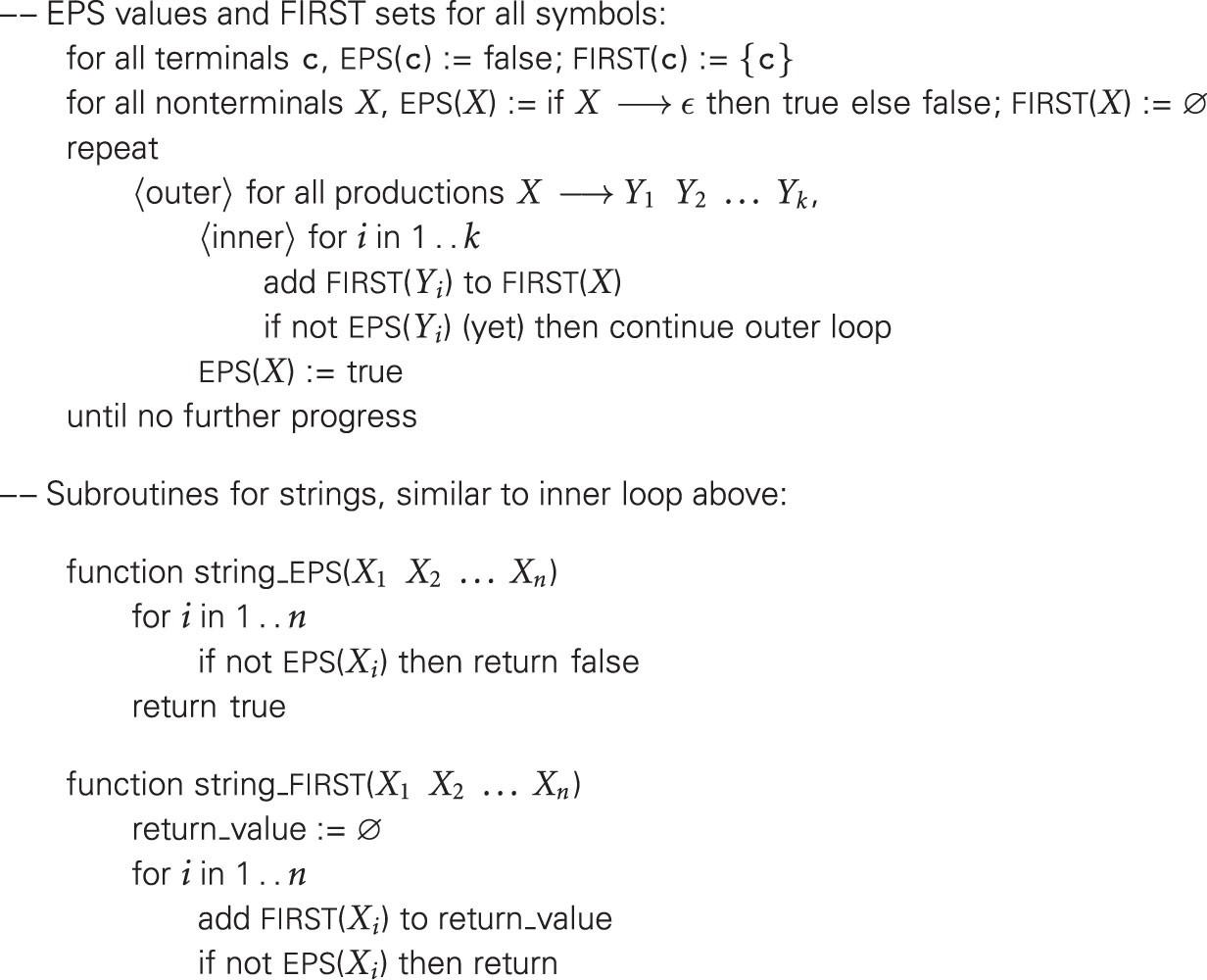
0

0 *≤ prod ≤* 9 is the production involved; indicate (0*, −*) when step

1. (5pt) For each production *i,* 1 *≤ i ≤* 9, compute predict(*i*).

is used for terminals.

1. (2pt) Using the information computed above, show that this grammar is not LL(1). (See definition on the slide 19 of the LR-parsing chapter.)
2. (10pt) Modify this grammar to make it LL(1). Explain clearly your changes and prove it is LL(1).



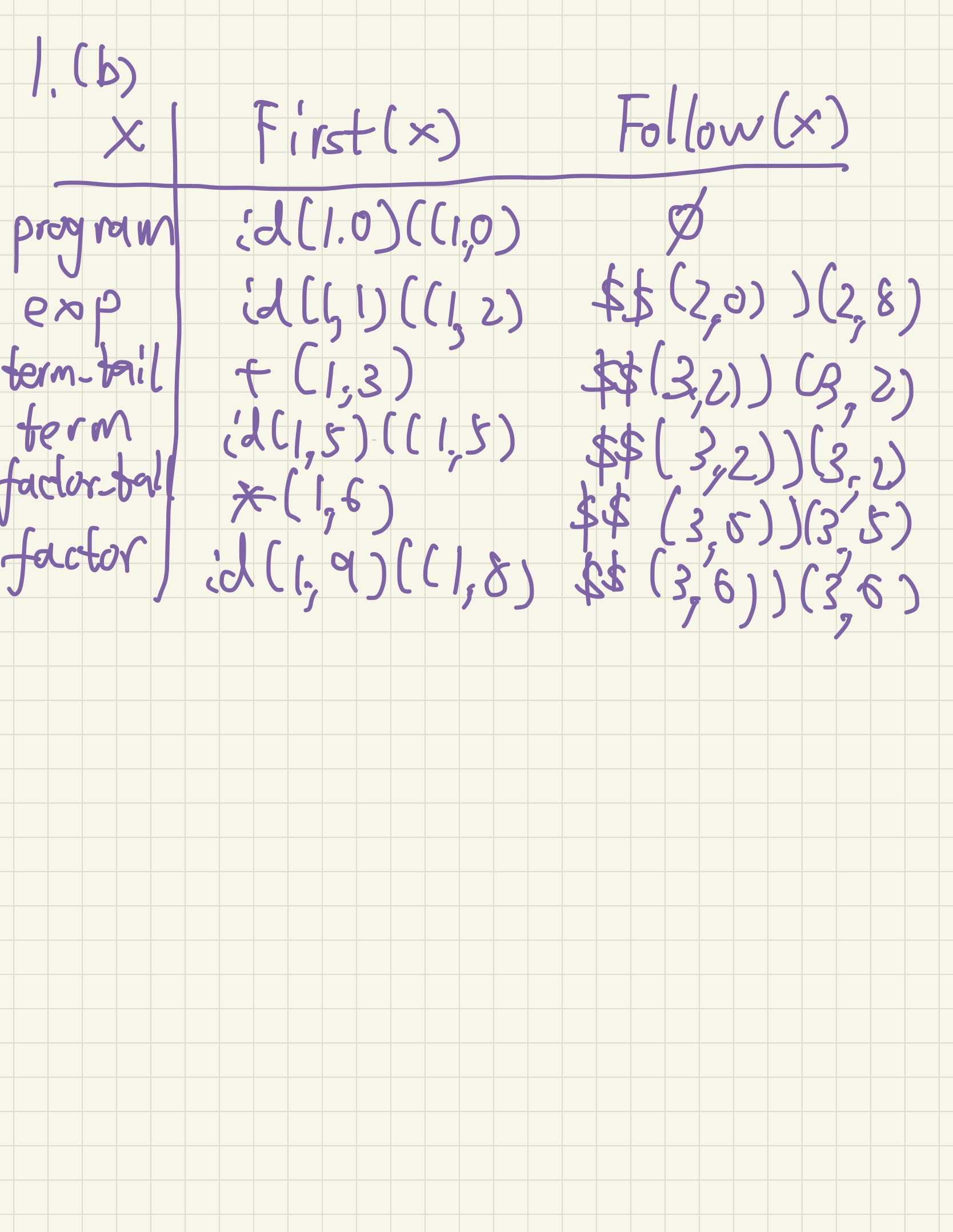
**0**

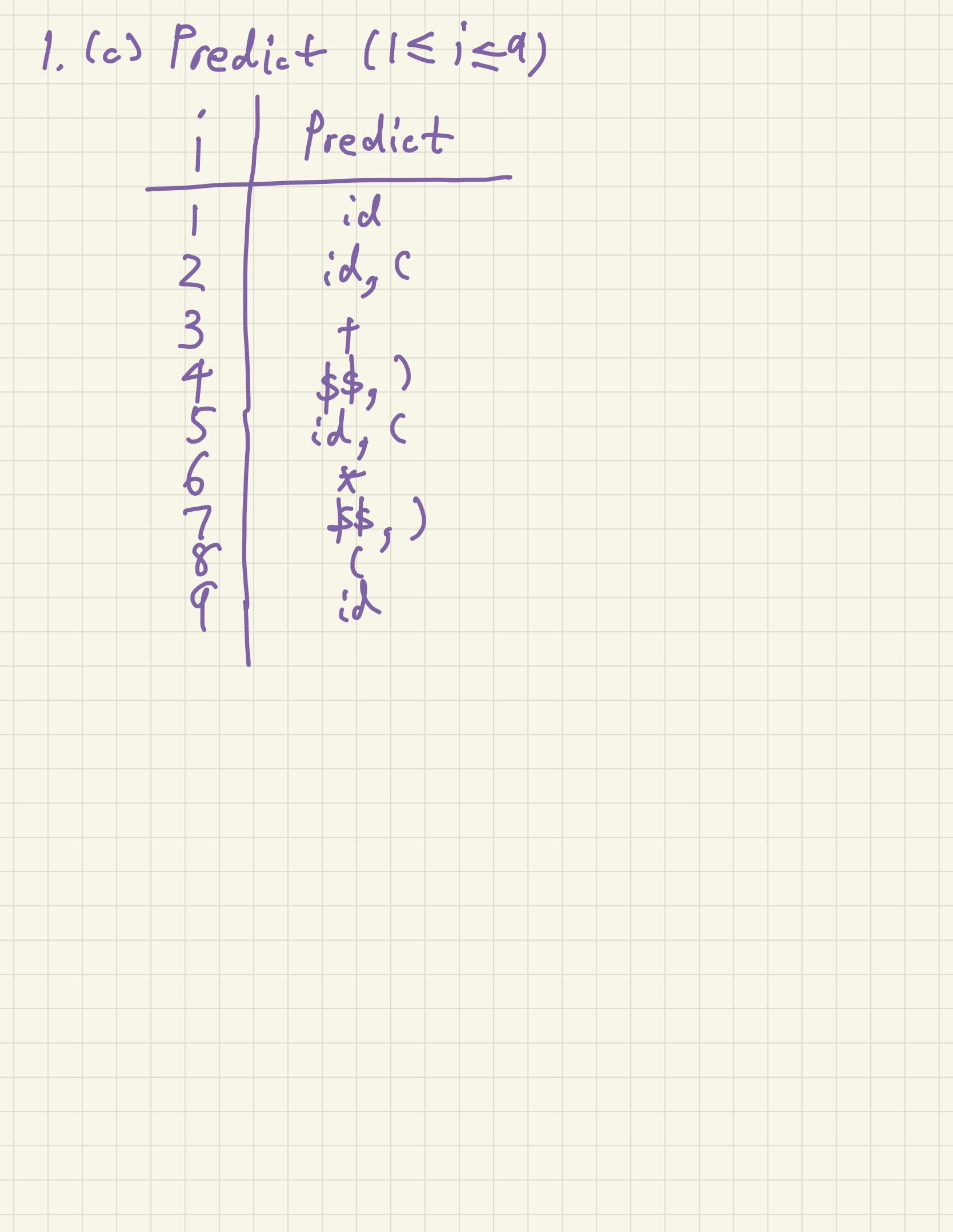
**1**

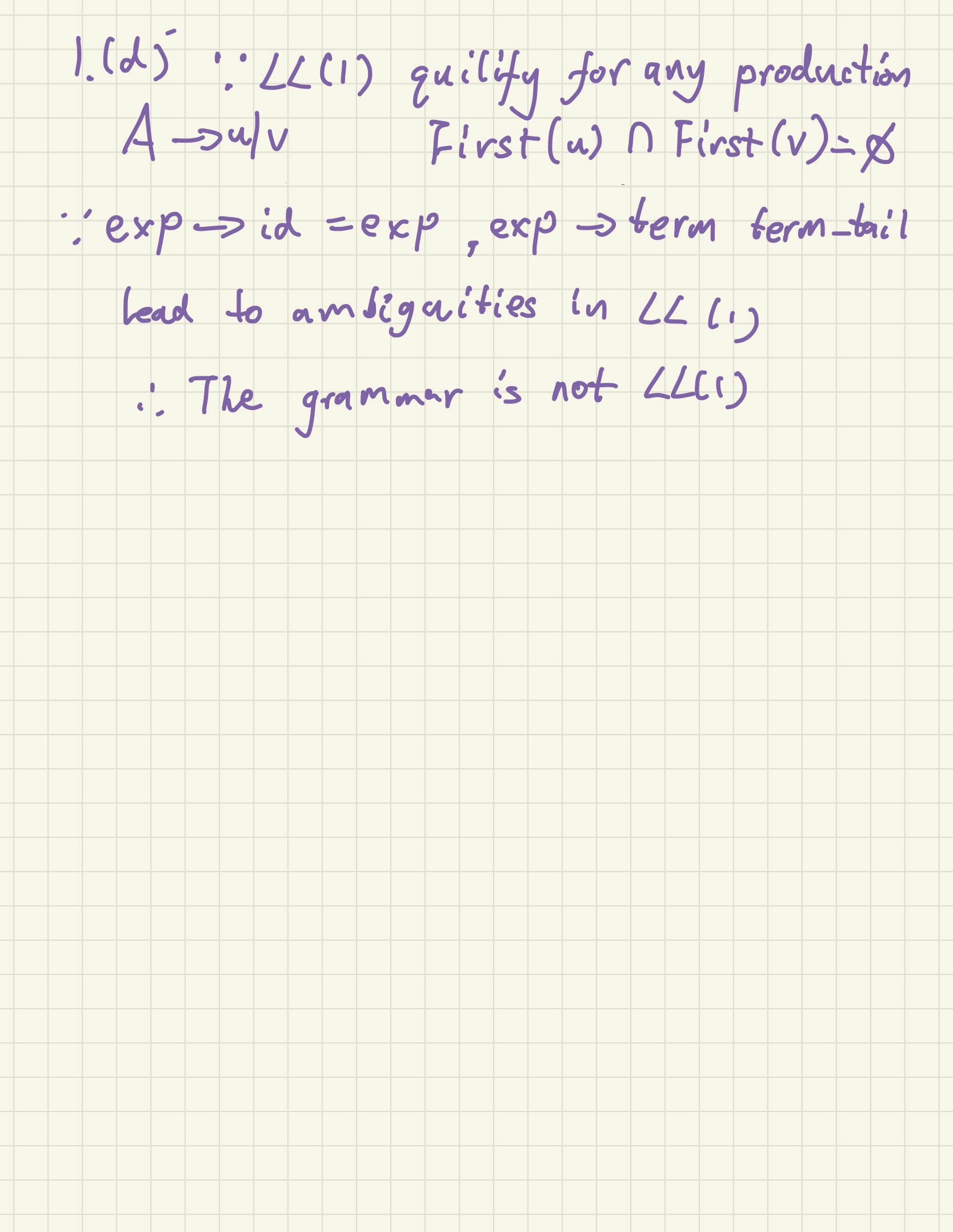
**2**

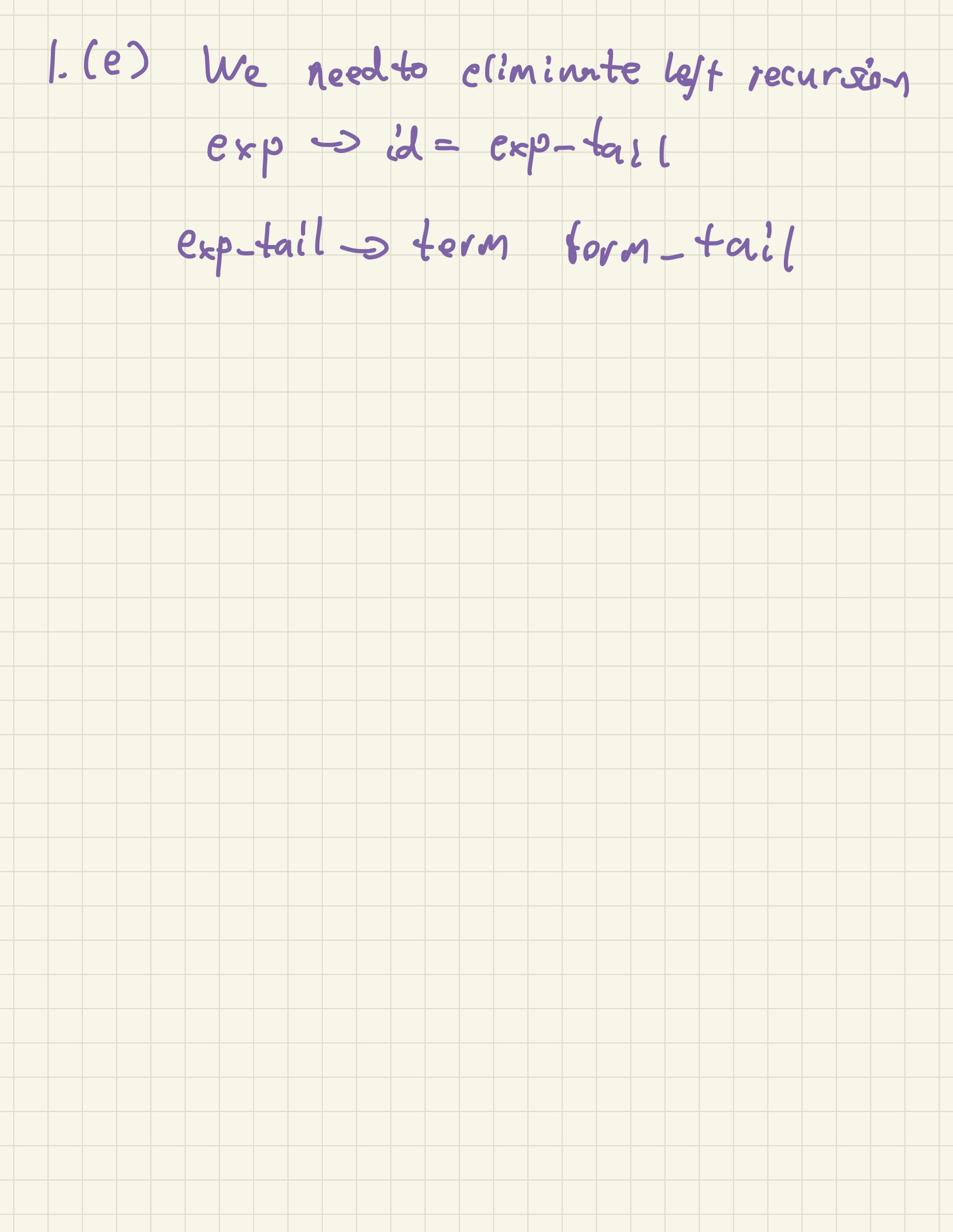
**3**



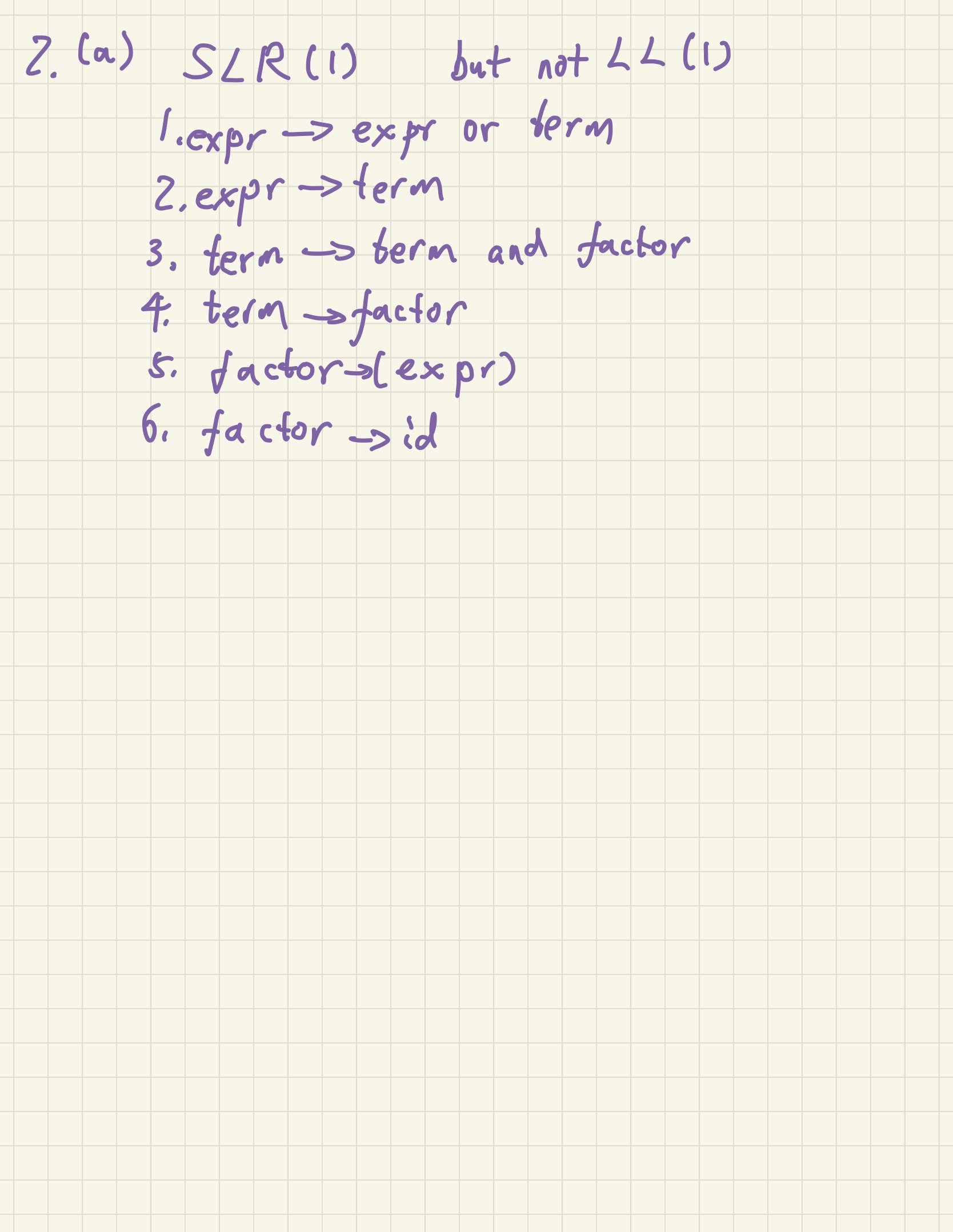


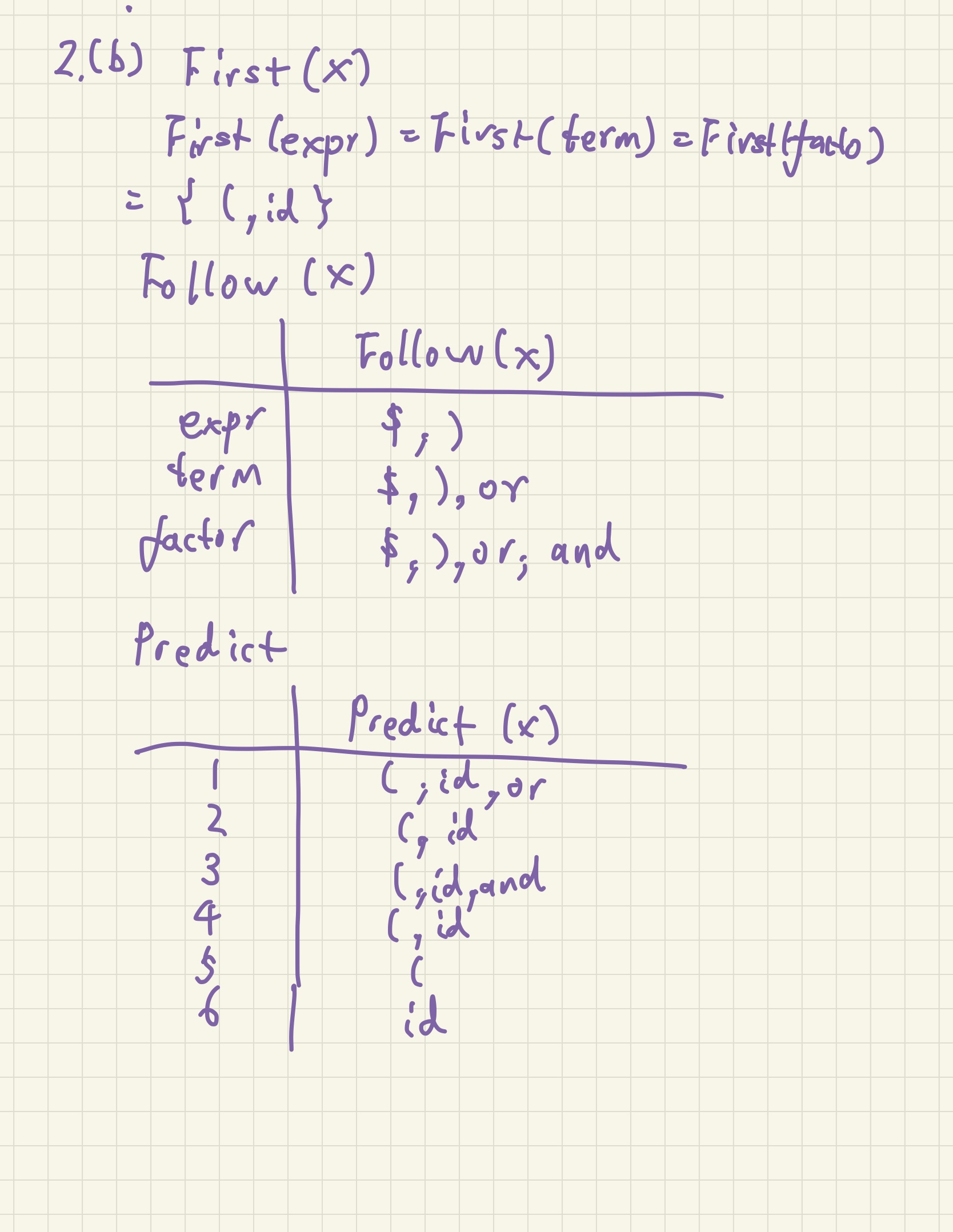






1. (30pt) Consider Boolean expressions containing operands (id), operators (and, or), and parentheses, where and has higher precedence than or.
2. (10pt) Write an SLR(1) grammar, *G*, which is not LL(1), for such expressions, which obeys the precedences indicated.
3. (5pt) Compute the first(*X*) and follow(*X*) sets for all nonterminals *X* and predict(*i*) sets for all productions *i*.
4. (5pt) Prove that *G* is not LL(1).
5. (10pt) Prove that *G* is SLR(1) by drawing the SLR graph and show there are no conflicts. Build the graph as shown in the examples we did in class (and done by jflap), not the condensed form in the textbook. For each state with potential conflicts (two LR-items, one with the dot in the middle, one with the dot at the end), explain clearly why there is no shift/reduce conflict.





1. (40pt) Consider the C-style switch statement.
2. (25pt) Write an S-attributed LL(1) grammar that generates C-style switch statements and checks that all labels of the arms of the switch instructions are distinct. In order to do that, the starting nonterminal, *S*, will have an attribute dup that will store all the duplicate values. There are no duplicate values on the arms of the switch statement if and only if *S*.dup = *∅*. Therefore, your grammar is required to eventually compute the attribute dup of *S*.

For simplicity, assume that the conditional expression of the switch statement and the constant expressions labelling the arms are expr tokens and that each arm has a statement that is a stmt token; the break and default parts are omitted as their role is irrelevant for our problem. Each expr has an attribute val provided by the scanner that gives the value of the expression.

Explain why your grammar works as required. For LL(1), you can use jflap to compute the parse table and show there is no conflict; include the jflap answer (whole window) in your answer.

1. (15pt) Using the above attributed grammar, draw a decorated parse tree for the following switch

instruction:

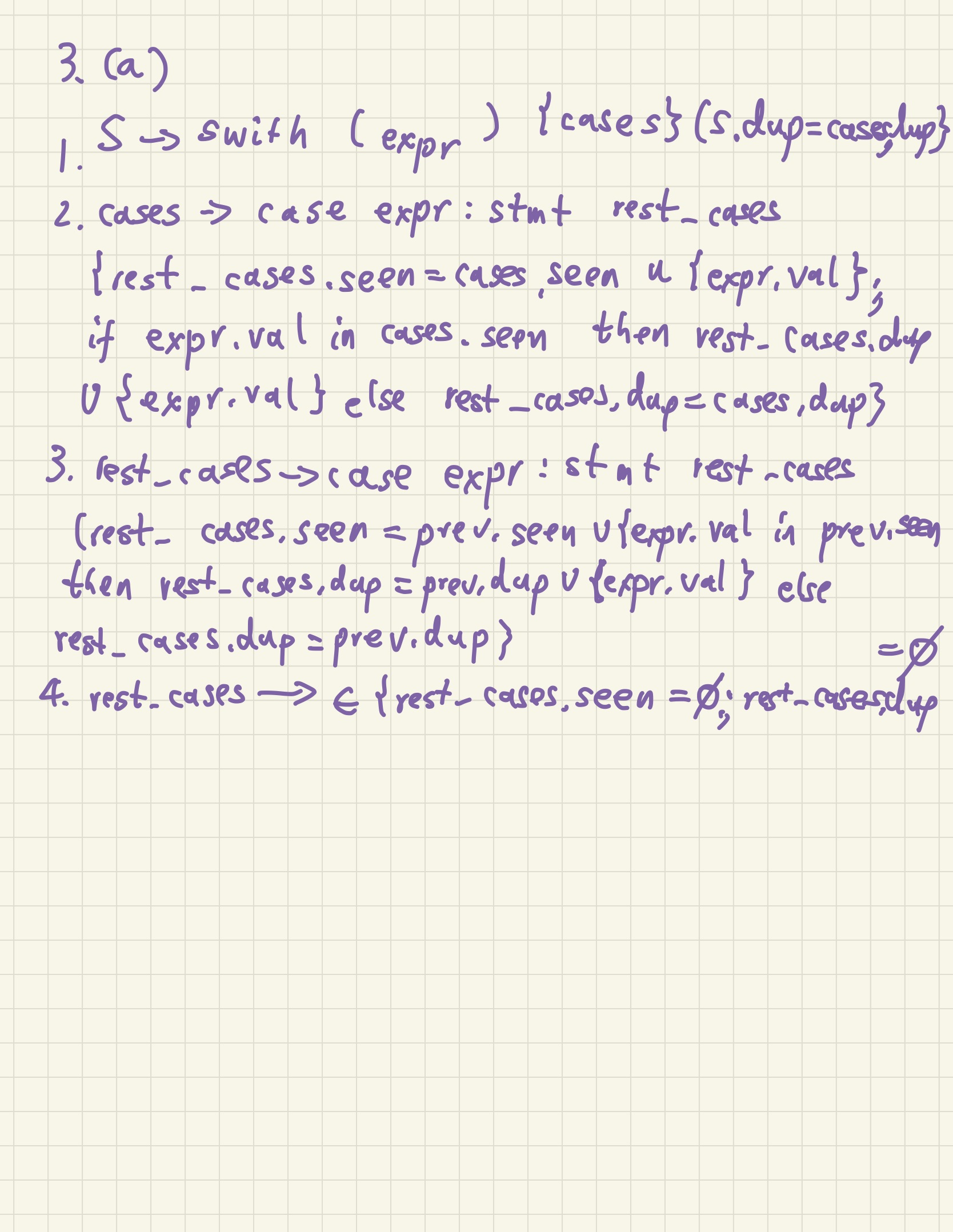
switch ( expr ) { case 2 : case 3 : stmt case 2 : stmt case 1 :

case 2 :

case 1: stmt

}

Show all attributes and arrows indicated what attributes are used to compute each value.



**Notes: JFLAP:** You are allowed to use JFLAP to help you solve the assignment. You still need to explain clearly your solution. Also, make sure you understand what it does; JFLAP will not be available during exams!

**LLMs:** You are allowed to use LLMs (Large Language Models), such as ChatGPT, but, again, they will not be available during exams.

**LATEX:** For those interested, the best program for scientific writing is LATEX. It is far superior to all the other programs, it is free, and you can start using it in minutes: [https://tobi.oetiker.ch/lshort/lshort.pdf.](https://tobi.oetiker.ch/lshort/lshort.pdf) It is also available online at [https://www.overleaf.com/.](https://www.overleaf.com/)