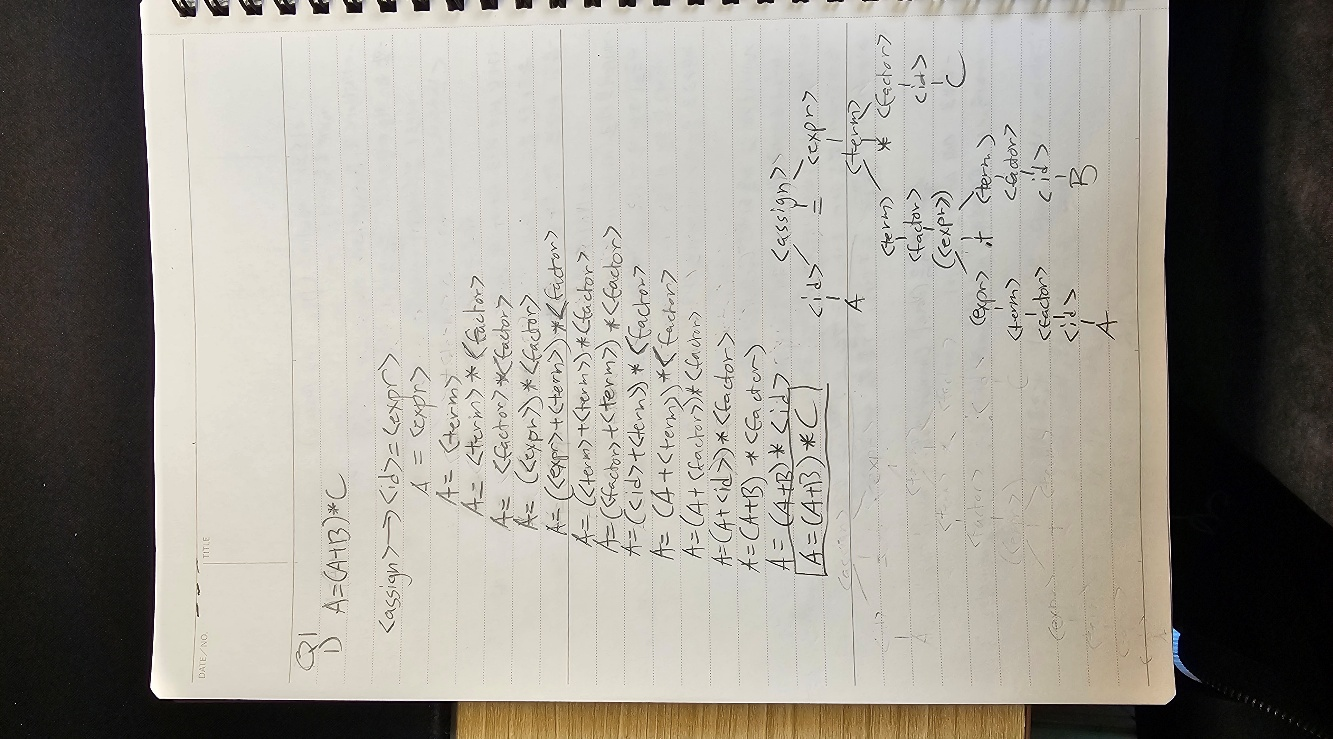
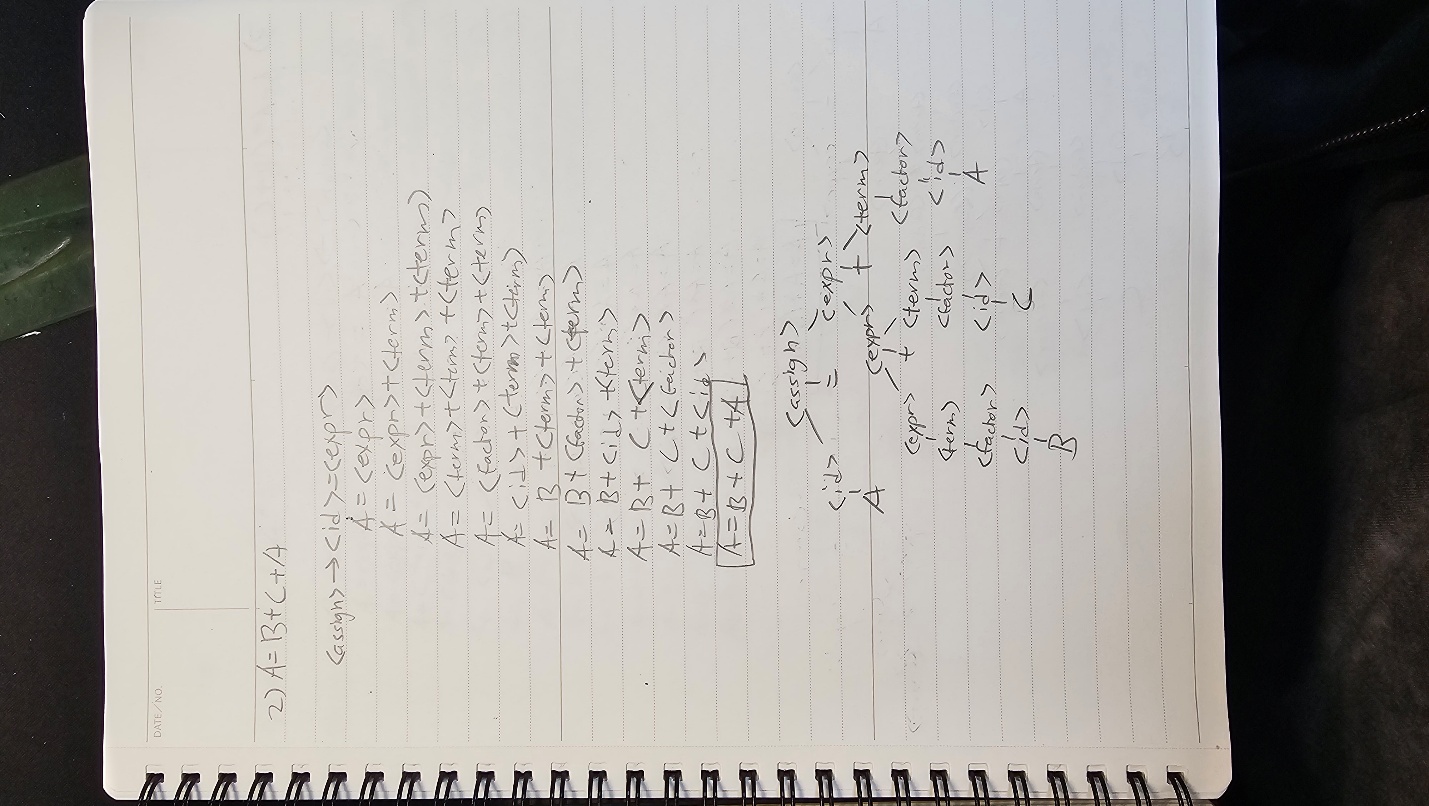
**Q1:** Using the BNF grammar below show a **parse tree** and a **leftmost derivation** for each of the following 4 statements:

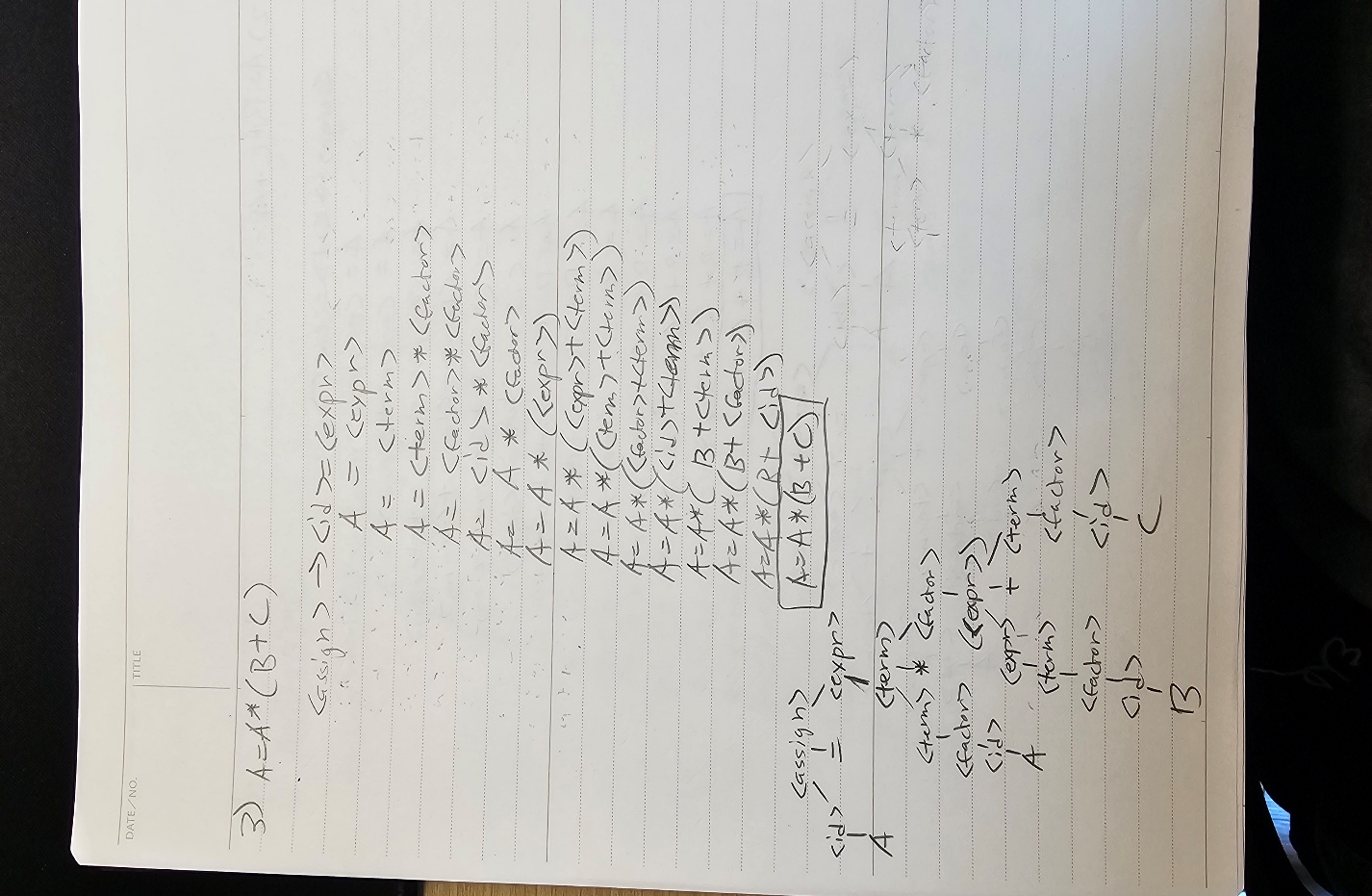
1. A = ( A + B ) \* C



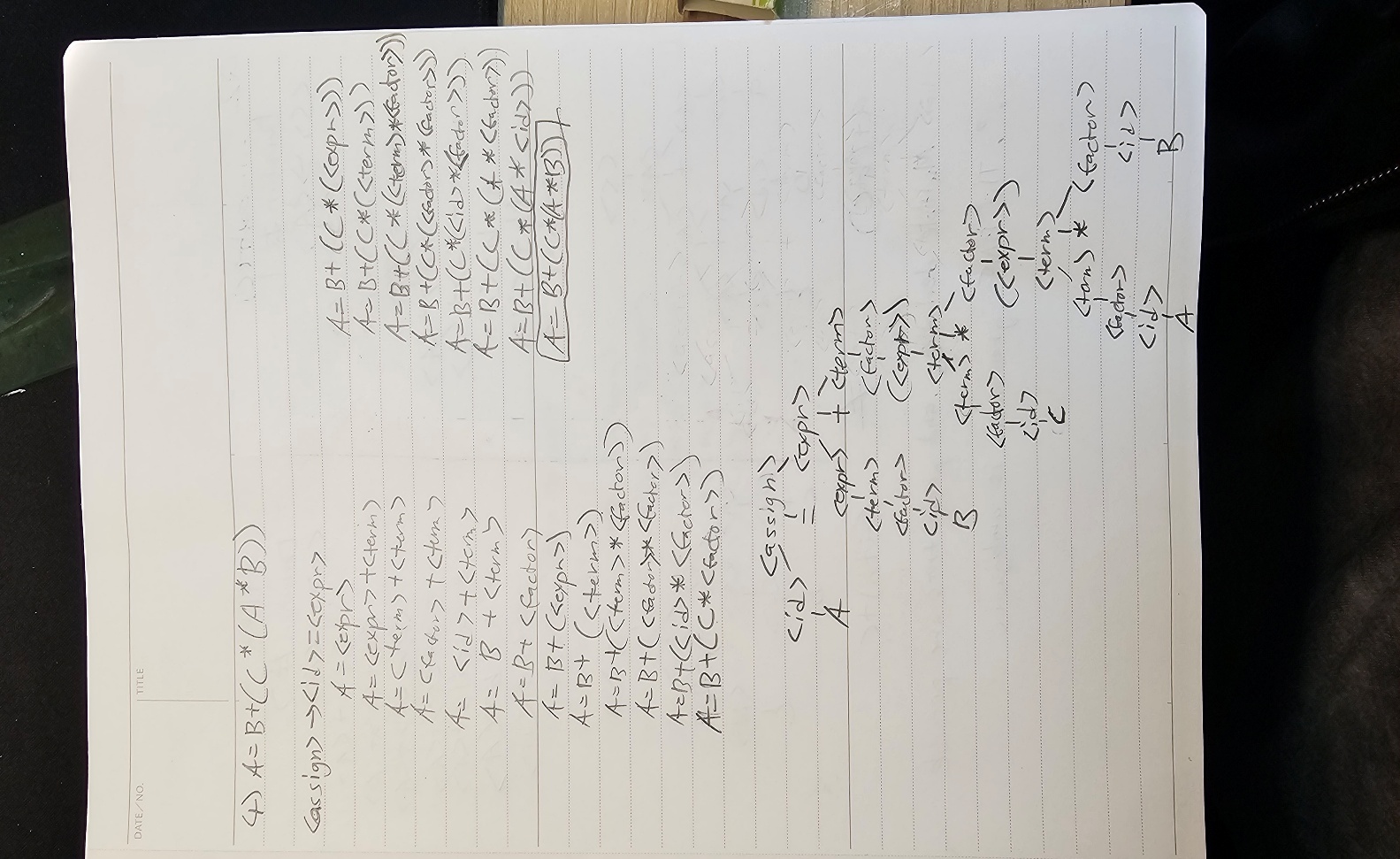
1. A = B + C + A



1. A = A \* (B + C)



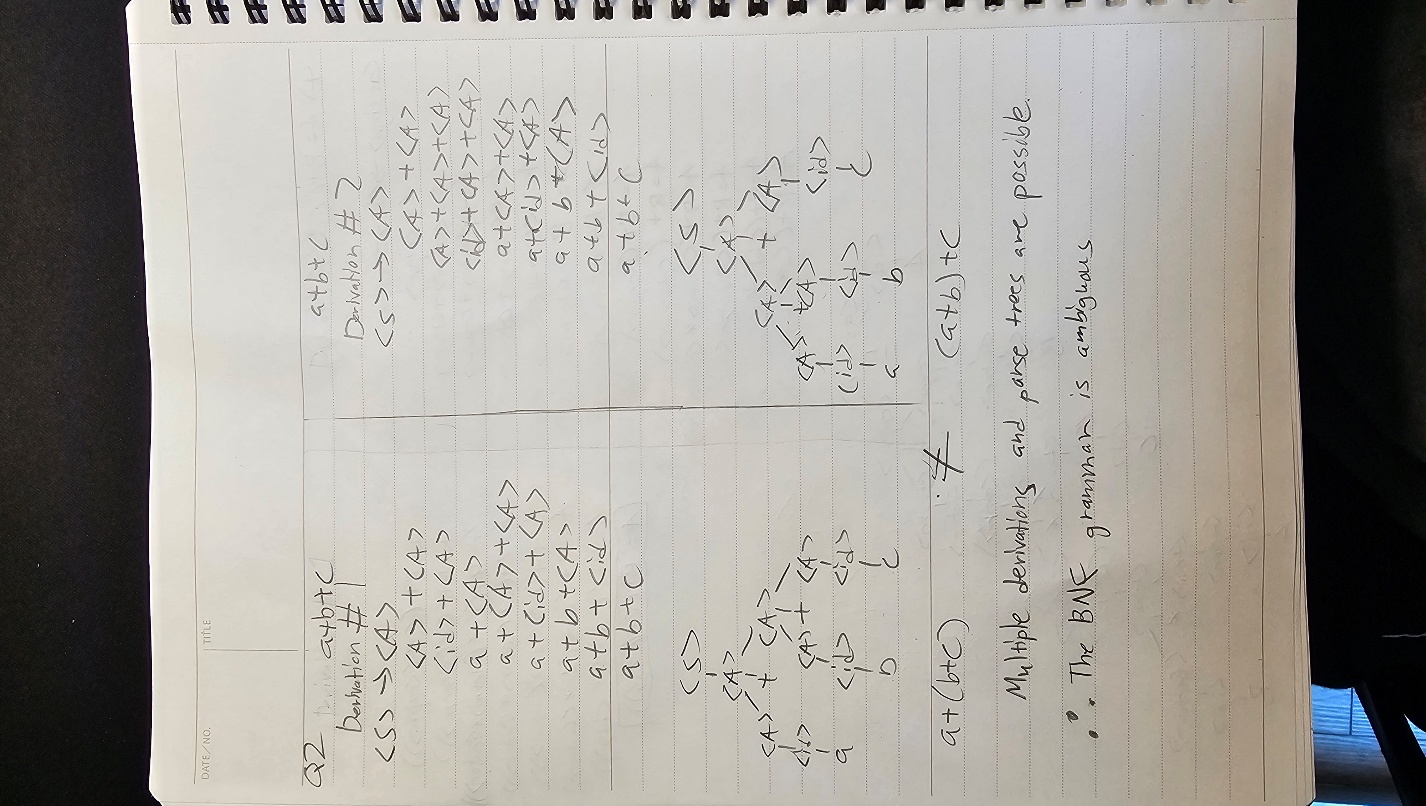
1. A = B + (C \* (A \* B))



|  |
| --- |
| BNF grammar |
| <assign> →<id> = <expr>  <id> A | B | C  <expr> →<expr> + <term> | <term>  <term> →<term> \* <factor> | <factor>  <factor> →( <expr> ) | <id> |

**Q2:** Prove that the following grammar is ambiguous:

|  |
| --- |
| BNF grammar |
| <S> →<A>  <A> →<A> + <A> | <id>  <id> →a | b | c |



**Q3:** Modify the grammar below to add a unary minus ─ operator and ^ power operator that have higher precedence than either + or \*.

Thus, the precedence of the operators in the final BNF should be ranked from the **highest** to the **lowest** as follows: ( ) , ─ , ^ , \* , +

Also, all the operators have left associativity except the power operator ^ and unary minus ─ , which have right associativity.

|  |
| --- |
| BNF grammar |
| <assign> →<id> = <expr>  <id> A | B | C  <expr> →<expr> + <term> | <term>  <term> →<term> \* <factor> | <factor>  <factor> →( <expr> ) | <id> |

