(12) (7 points) Write EBNF and syntax graph descriptions for the following:
(i). A Java class definition header statement
(ii). A C switch statement

**Ans.**

(i) \(<class\_head> \to \{ <modifier>\} class <id> [extends class\_name] [implements <interface\_name> {, <interface\_name>}]<n><modifier> \to public | abstract | final

(ii) \(<switch\_stmt> \to switch ( <expr> ) \{ case <literal> : <stmt\_list> \{ case <literal> : <stmt\_list> \} [default : <stmt\_list>] \}

(13) (7 points) Rewrite the BNF of Example 3.4 to give + precedence over * and force + to be right associative.

**Ans.**

\(<assign> \to <id> = <expr><n><id> \to A \mid B \mid C\n
\(<expr> \to <expr> \star <term><n> <term> \to <factor> + <term><n> <factor> \to ( <expr> ) \mid <id><n><factor> \to <id> | - <id>

(14) (7 points) Modify the grammar of Example 3.4 to add a unary minus operator that has higher precedence than either + or *.

**Ans.**

Assume that the unary operators can precede any operand.

Replace the rule

\(<factor> \to <id>\)

with

\(<factor> \to <id> | - <id>\)
Write an attribute grammar whose BNF basis is that of Example 3.6 in Section 3.4.5, but whose language rules are as follows: Data types cannot be mixed in expressions, but assignment statements need not have the same types on both sides of the assignment operator.

Ans.

1. Syntax rule: `<assign> -> <var> = <expr>`
   predicate: `<var>[2].actual_type = <var>[3].actual_type`
3. Syntax rule: `<expr> -> <var>`
4. Syntax rule: `<var> -> A | B | C`
   Semantic rule: `<var>.actual_type <- look-up(<var>.string)`