LOAL Basic Terminology — Exercises

1. **Vocabulary, Grammar, Syntax Tree**

2. **Classical Notation → EBNF**

   Given the grammar in classical notation for a simple arithmetic expression in a programming language:

   Expression = Term ExpressionPrime
   ExpressionPrime = AddOp Term ExpressionPrime | ε
   Term = Factor TermPrime
   TermPrime = MulOp Factor TermPrime | ε
   Factor = Number | Identifier | "(" Expression ")"
   AddOp = "+" | "-"
   MulOp = "*" | "/"

   (a) Which symbols have to be in V_N?

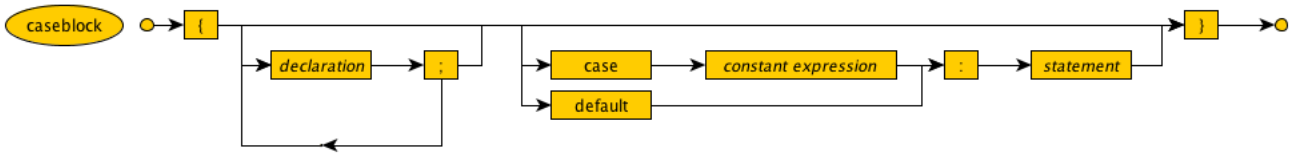
   (b) Rewrite this grammar in EBNF.

   (c) Draw the syntax trees for the string `17 * (a + 20 * b) - c`. Use the classical notation for the syntax tree first and the EBNF notation you developed as answer for question b second.

   Solution: The classical notation for the given syntax diagram is:

   Expression = Term { AddOp Term }
   Term = Factor { MulOp Factor }
   Factor = number |identifier |"(" Expression ")"
   AddOp = "+" |"-"
   MulOp = "*" |"/"

3. **Syntax Diagram → Classical Notation** Given the following rule of grammar for the `switch` Statement in C in form of a syntax diagram.



   Rewrite this rule in the form of classical formal language grammar notation.

   Gegeben sei obiges Syntax Diagramm. Schreiben Sie dieses in die klassische Grammatik-Notation um.

4. **Java Method Definition** Give the grammar of the Java method definition in EBNF. You may use the non-terminals "Statement", "Type", and "Identifier". Split the grammar into useful non-terminals. MethodDefinition, MethodHead, MethodBody, ParameterList, OneParameter could be useful candidates.

   Geben Sie die Grammatik für Methodendefinition in JAVA in EBNF an. Sie dürfen die Non-Terminal-Symbole "Statement", "Type" und "Identifier" als gegeben voraussetzen und brauchen diese nicht weiter zu definieren. Teilen Sie Ihre Grammatik in vernünftige Non-Terminale auf. MethodDefinition, MethodHead, MethodBody, ParameterList, OneParameter könnten vernünftige Kandidaten sein.

5. **Simple Protocol** A simple protocol to transmit data from sender to a receiver has a header followed by one or more data fragments. The header contains a version string which starts with the character "V" followed by a max. two-digit number. Optionally an arbitrary number of further two digits number may follow (sub-version identifier) which are separated by "." (dots). The string `V2.01.4.00` would be a valid version string. After this an optional sender address may follow. The header is finished by the receiver's address. Both addresses are 6 digit numbers, where the sender address is pre-fixed by an "S" and the receiver address is pre-fixed by a "D".

   A data fragment starts with a max. three digit number denoting the size of the following data. The data itself is a sequence of one or more characters (obviously as many as given in the data size field before). The data fragment is finished by a flag "T" or "F" indicating whether a next data fragment follows or not.

   Describe this protocol in form of Wirth's EBNF. You may assume to have the terminal classes "digit" and "char".

   Ein einfaches Protokoll zur Datenübertragung von einem Sender zu einem Empfänger besteht aus einem Header und einem oder mehreren Datenfragmenten. Der Header startet mit einem Versionsstring, der mit dem Zeichen "V" beginnt und von einer maximal zweistelligen Zahl gefolgt wird. Eventuell können noch weitere max. zweistellige Zahlen folgen (Bezeichner für Unterversionen), welche durch einen "." (Punkt) voneinander getrennt sind. Der String `V2.01.4.00` wäre ein gültiger Versionsstring. Nach dem Versionsstring folgt optional eine Senderadresse. Der Header wird von der Empfängeradresse abgeschlossen. Beide Adressen sind sechsstellige Zahlen, wobei die Senderadresse ein "S" und die Empfängeradresse ein "D" vorangestellt hat.

   Ein Datenfragment startet mit einer maximal dreistelligen Zahl, welche die Anzahl der Zeichen der nun folgenden Daten beschreibt. Die Daten selbst sind eine Sequenz von einem oder mehreren Zeichen (characters), offensichtlich so viele, wie vorher bei der Anzahl angegeben wurden. Das Datenfragment wird von einem Flag "T" oder "F" abgeschlossen, welches anzeigt, ob noch ein weiteres Fragment folgt oder nicht.

   Beschreiben Sie dieses Protokoll mit Hilfe der Wirth'schen EBNF. Sie dürfen die Terminalklassen "digit" und "char" voraussetzen.

Solution:

Message	=	Header DataFragments.
Header	=	Version TotalLength [SourceAddress] DestinationAddress.
Version	=	"V"digit [digit] {"." digit[digit]}.
SourceAddress	=	"S" Address.
DestinationAddress	=	"D" Address.
Address	=	digit digit digit digit digit digit.
DataFragments	=	DataFragment {DataFragment}.
DataFragment	=	DataSize Data IsLastFragment.
DataSize	=	digit [digit [digit]].
Data	=	char {char}.
IsLastFragment	=	"T" | "F".