Exercise 06 - Chomsky-Normal Form

Any context-free grammar can be converted to Chomsky Normal Form, which is a CFG (Q, Σ, R, S) such that every production is either in the form

- $A \rightarrow BC$
- $A \rightarrow a$

where $A, B, C \in Q$ and $a \in \Sigma$. Furthermore, it includes the rule $S \rightarrow \varepsilon$ only if S is not in any substitution of R. To convert a CFG into CNF, you need to perform four steps:

1. Create a new start variable, S_0 and add rule $S_0 \rightarrow S$ to R.
2. Remove all ε-rules, $A \rightarrow \varepsilon$ [excluding the start variable ε-rule].
3. Remove all unit rules, $A \rightarrow B$, where $A, B \in Q$.
4. Construct lexical rules, $A \rightarrow a$, and binary rules, $A \rightarrow BC$, where $A, B, C \in Q$ and $a \in \Sigma$.

The algorithms for each step are in the lecture notes.

Your objective is to convert a grammar into Chomsky Normal Form (CNF) using five separate transformation functions. The grammar is represented as a set of production rules, where the domain of the first rule is designated as the start variable.

Each rule is defined by the class *Rule*, which consists of a character–string pair (domain–substitution pair). The domain must be a single uppercase letter, and the substitution must be a string containing only letters, digits, and the period character (.), which represents ε (the empty string).

Create a C++ file named '`exercises06.cpp`' that includes the header file '`Utility.h`' and defines the required five functions. Each function must:

- Accept a reference to a *Rule Set* as a parameter
- Accept a string of available variables as a parameter
- Return a Boolean (true if the transformation was successful; otherwise, false)

The functions are:

1. Define `NewStart()` that performs task 1 of the CNF conversion process.
2. Define `RemoveEpsilons()` that performs task 2 of the CNF conversion process.
3. Define `RemoveUnits()` that performs task 3 of the CNF conversion process.
4. Define `ConstructLexicals()` that performs the first half of task 4 of the CNF conversion process.
5. Define `ConstructBinarys()` that performs the second half of task 4 of the CNF conversion process.