1. If you routinely build electronic circuits that perform logic operations, you may want to buy a supply of circuits that implement miscellaneous operators, or gates, as they are usually called in the business. Everything can be built out of “not” (¬), “and” (∧), and “or” (∨) gates. For instance, to make an implication \( p \rightarrow q \), we use the following logical equivalence:

\[
p \rightarrow q \equiv \neg p \lor q
\]

(if you are not sure, check with a truth table that this equivalence holds).

However, this poses a dilemma: how many gates of each type should you buy without knowing what circuits you are going to build? The industry came up with a clever solution: just buy many nand gates. The following is the truth table for the nand of \( p \) and \( q \), denoted \( p \mid q \):

\[
\begin{array}{ccc}
p & q & p \mid q \\
T & T & F \\
T & F & T \\
F & T & T \\
F & F & T \\
\end{array}
\]

(a) Write four logical equivalences that let you implement negation (¬), conjunction (\( p \land q \)), disjunction (\( p \lor q \)), and implication (\( p \rightarrow q \)) with “nand” gates (\( \mid \)). Your equivalences will have the following form:

\[
\begin{align*}
\neg p & \equiv \ldots \\
p \land q & \equiv \ldots \\
p \lor q & \equiv \ldots \\
p \rightarrow q & \equiv \ldots \\
\end{align*}
\]

where the only operators on the right-hand sides are nands, (you’ll also need parentheses).

(b) Is the nand associative? Prove your answer.

2. Express the following sentence in predicate logic:

Some people like all food, but no food is liked by everybody.

In doing so, use the predicate \( L(x,y) \) meaning “person \( x \) likes food item \( y \).”
3. Let $R$ be the following inference rule:

\[
\frac{\phi \lor \psi, \psi \rightarrow \omega}{\phi \lor \omega}.
\]

This is the only inference rule at your disposal.

(a) Prove that $R$ is sound. Be clear and specific in your argument.

(b) You are given this knowledge base in propositional logic:

\[
p \lor q, p \rightarrow r, \neg q
\]

Use $R$ (repeatedly) to prove that $r$ is true. You will need to use logical equivalences. In particular, you may also need the obvious equivalence

\[
\phi \equiv \phi \lor \neg \phi
\]

where $\neg \phi$ is the contradiction. Show all your steps clearly.

4. Consider the following statement

I won’t come with you unless you let me drive.

(a) Let $p$ be the proposition “I’ll come with you” and let $q$ be the proposition “You let me drive.” Write the statement above as a formula in propositional logic.

(b) Write the inverse of the formula you wrote to express the given statement, and translate your new formula to English.

(c) Write the converse of the formula you wrote to express the given statement, and translate your new formula to English.

(d) Write the contrapositive of the formula you wrote to express the given statement, and translate your new formula to English.

(e) Which of the three transformations (inverse, converse, contrapositive) is logically equivalent to the original statement? Prove it by truth table.

5. A few examples of Backus-Naur Form (BNF) grammars in order of increasing difficulty.

(a) Write a BNF grammar for the language that contains exactly all strings that have a nonnegative number of $x$ symbols followed by an equal number of $y$ symbols. Zero $x$-es followed by zero $y$-s form the empty string, which is part of the language. Use the symbol $\emptyset$ to denote the empty string. For instance, $\emptyset$, $xy$, $xxyyy$ are strings in the language, but $x$, $xxyy$, $xxxx$ are not.

(b) Write a BNF grammar for the language that contains exactly all strings that have $m$ $x$ symbols (with $m \geq 0$) followed by $n$ $y$ symbols (with $n \geq 0$), followed by $m + n$ $z$ symbols. For instance, $\emptyset$, $xyz$, $yyzzz$, $xzz$, $xxyyyyy$ are strings in the language, but $xxyzz$, $yyyyyyyyyyyyyyyy$ are not. Try to use meaningful names for non-terminal symbols.

(c) Write a BNF grammar for the language that contains exactly all strings that have a nonnegative number of $x$ symbols followed by a different number of $y$ symbols. For instance, $x$, $y$, $xxy$, $xxyyyyy$ are strings in the language, but $\emptyset$, $xxxxyy$, $yyyyx$ are not. [Hint: a string in this language has either more $x$ symbols than $y$ symbols, or vice versa . . .]