Can we express \( \rightarrow \) using just \( \neg, \land, \lor \)?

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

If \( \neg p \), \( p \rightarrow q \) is \( T \) otherwise \( p \rightarrow q \) is equivalent to

\[
\text{so, } p \rightarrow q \equiv \neg p \lor q
\]
contrapositive of \( p \rightarrow q \) is \( \neg q \rightarrow \neg p \)

converse of \( p \rightarrow q \) is \( q \rightarrow p \)

inverse of \( p \rightarrow q \) is \( \neg p \rightarrow \neg q \)
"If the AC is on, then I'll be cold."

**Contrapositive:**
"If I'm not cold, the AC is not on."

**Converse:**
"If I'm cold, the AC is on."

**Inverse:**
"If the AC is not on, then I won't be cold."
Prove: an implication and its contrapositive are logically equivalent.

i.e., \( p \rightarrow q \equiv \neg q \rightarrow \neg p \)

<table>
<thead>
<tr>
<th>( p )</th>
<th>( q )</th>
<th>( p \rightarrow q )</th>
<th>( \neg p )</th>
<th>( \neg q )</th>
<th>( \neg q \rightarrow \neg p )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
\( p \leftrightarrow q \) expresses the biconditional statement "p if and only if q", which is equivalent to the proposition \((p \rightarrow q) \land (q \rightarrow p)\)

```

truth table:

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>P \leftrightarrow Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
```

"iff"
The biconditional is often implied by the English "if". E.g., "if you brush your teeth, then you can go to bed."

(if the biconditional is not implied, what loophole exists?)
When propositions are logically equivalent, we can substitute one for the other in statements they appear in — this can be useful in many ways (e.g. simplification/reduction, standardization).

\[ p \equiv q \text{ (p is equivalent to q) if } p \leftrightarrow q \text{ is a tautology.} \]
Show that \( \neg(p \lor q) \equiv \neg p \land \neg q \).
### TABLE 6  Logical Equivalences.

<table>
<thead>
<tr>
<th>Equivalence</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p \land T \equiv p )</td>
<td>Identity laws</td>
</tr>
<tr>
<td>( p \lor F \equiv p )</td>
<td>Domination laws</td>
</tr>
<tr>
<td>( p \lor T \equiv T )</td>
<td>Idempotent laws</td>
</tr>
<tr>
<td>( p \land F \equiv F )</td>
<td>Double negation law</td>
</tr>
<tr>
<td>( p \lor q \equiv q \lor p )</td>
<td>Commutative laws</td>
</tr>
<tr>
<td>( p \land q \equiv q \land p )</td>
<td>Associative laws</td>
</tr>
<tr>
<td>( p \lor (q \land r) \equiv (p \lor q) \land (p \lor r) )</td>
<td>Distributive laws</td>
</tr>
<tr>
<td>( p \land (q \lor r) \equiv (p \land q) \lor (p \land r) )</td>
<td>De Morgan’s laws</td>
</tr>
<tr>
<td>( p \lor (p \land q) \equiv p \lor (q \land r) )</td>
<td>Absorption laws</td>
</tr>
<tr>
<td>( p \land (p \lor q) \equiv p \lor (q \land r) )</td>
<td>Negation laws</td>
</tr>
</tbody>
</table>

### TABLE 7  Logical Equivalences Involving Conditional Statements.

\[
p \rightarrow q \equiv \neg p \lor q \\
p \rightarrow q \equiv \neg q \rightarrow \neg p \\
p \lor q \equiv \neg p \rightarrow q \\
p \land q \equiv \neg(p \rightarrow \neg q) \\
\neg(p \rightarrow q) \equiv p \land \neg q \\
(p \rightarrow q) \land (p \rightarrow r) \equiv p \lor (q \land r) \\
(p \rightarrow r) \land (q \rightarrow r) \equiv (p \lor q) \rightarrow r \\
(p \rightarrow q) \lor (p \rightarrow r) \equiv p \lor (q \lor r) \\
(p \rightarrow r) \lor (q \rightarrow r) \equiv (p \land q) \rightarrow r \\
\]

### TABLE 8  Logical Equivalences Involving Biconditional Statements.

\[
p \leftrightarrow q \equiv (p \rightarrow q) \land (q \rightarrow p) \\
p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q \\
p \leftrightarrow q \equiv (p \land q) \lor (\neg p \land \neg q) \\
\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q \\
\]
Predicate logic
(aka First-Order Logic)
Predicate logic adds variables, predicates, and quantifiers to propositional logic.

e.g. propositional logic:
   p: “John likes cake.” q: “Jane likes cake”

   e.g. predicate logic:
   P(x): “x likes cake”
   \( \exists x \) P(x): “there exists x such that x likes cake”
A predicate is a statement that is True or False, depending on the value of one or more variables taken from the domain or "universe of discourse."

A predicate $P(x,y,\ldots)$ can be thought of as a propositional function that evaluates to $T/F$, based on its inputs $x,y,\ldots$
e.g. \( P(x) \): \( x < 10 \) and \( x \) is prime

\[
\begin{align*}
P(3) &?: T \\
P(9) &?: F \\
P(11) &?: F \\
P(20) &?: F \\
P(5) \lor P(20) &?: T \\
P(7) \land P(10) &?: F \\
P(1) \rightarrow P(12) &?: F \\
\end{align*}
\]
e.g. $Q(x, y, z) : x + y = z$

$Q(1, 2, 3) \Rightarrow T$

$Q(3, -1, 4) \Rightarrow F$

$Q(2, 4, 7) \Rightarrow Q(3, 4, 7) \Rightarrow T$

$F \Rightarrow T$
A quantifier specifies how many values from the domain, when assigned to a particular variable, satisfy a predicate.

Most important quantifiers:
- universal quantifier: $\forall$ — "for all"
- existential quantifier: $\exists$ — "there exists"
Eq. 1

\forall x \ P(x) : "For all \ x \ in \ the \ domain, \ P(x) \ is \ True."

\exists x \ P(x) : "There \ exists \ some \ x \ in \ the \ domain 
\hspace{1cm} such \ that \ P(x) \ is \ True."

the quantifiers bind the variable \ x
\( \forall x \, P(x) \) : "For all \( x \) in the domain, \( P(x) \) is True."

\( \exists x \, P(x) \) : "There exists some \( x \) in the domain such that \( P(x) \) is True."

Note: Important to specify the domain!

(we'll discuss different ways of doing this)
Quantifiers can be thought of as looping over values in the domain.

- $\forall x \, P(x)$ loops over all $x$ in domain
  - only true if $P(x)$ is true for all iterations
  - if any $x$ causes $P(x)$ to be false, terminate with false

- $\exists x \, P(x)$ loops over all $x$ in domain
  - if we find $P(x)$ true for some $x$, terminate with true
  - if loop terminates without finding $P(x)$ true for any $x$, evaluates to false
Quantifiers can also be considered in terms of logical conjunctions/disjunctions (for finite domains)

- $\forall x P(x)$ is the logical conjunction of $P(x)$ for all $x$
  
  i.e., $\forall x P(x) \equiv \bigwedge_x P(x)$

- $\exists x P(x)$ is the logical disjunction of $P(x)$ for all $x$
  
  i.e., $\exists x P(x) \equiv \bigvee_x P(x)$
On Precedence

∀ and ∃ have higher precedence than all logical operators!

E.g. \( \forall x \, P(x) \lor Q(x) \equiv (\forall x \, P(x)) \lor Q(x) \)

\( \forall x \, P(x) \lor Q(x) \neq \forall x \, (P(x) \lor Q(x)) \)
We can come up with other quantifiers...

e.g., "there are exactly N values ..."

"for the majority of values ..." (assuming finite domain)

"there is a unique value ..."

but we can express most other quantifiers using propositional operators.
Uniqueness Quantifier "\( \exists! \)"

\( \exists! x \; P(x) \) : "There is a unique \( x \) such that \( P(x) \)."

e.g. \( P(x) : x + 10 = 0 \), domain is \( \mathbb{Z} \)

\( \exists! x \; P(x) \) ? \( \text{T} \)

e.g. \( P(x) : x < 0 \), domain is \( \mathbb{Z} \)

\( \exists! x \; P(x) \) ? \( \text{F} \)
Express $\exists!$ in terms of $\exists$ and $\forall$:

$\exists! x \neg P(x) \equiv \exists x (P(x) \land \forall y (P(y) \rightarrow y = x))$