This is a tutorial on a proof strategy for a common programming pattern. You have probably written (pieces of) programs in the following form:

(initialization).
while ¬(done)
doin (one step) od

In this course we require specifications: one for the whole program, and one for the loop. Let's call them $P$ and $Q$. So we have two refinements.

$P \iff$ (initialization). $Q$
$Q \iff$ if (done) then ok else (one step). $Q$ fi

Specification $P$ describes the problem we are solving. Usually that's given, maybe informally, and our job is to formalize it. We have to invent $Q$, and that's probably the hardest part of the whole exercise. Often, $Q$ is a lot like $P$. $P$ describes the whole problem, and $Q$ describes what's still to be done when we're somewhere in the middle of execution.

As a simple example, $P$ may talk about all the items of a list, indexes $0,..#L$, and we are processing them in order. In the middle of execution, the remaining items are $k,..#L$ for some variable $k$, so that's how we change $P$ into $Q$. Or maybe we are accumulating a sum, so $P$ says $s'=s+\Sigma$(some terms). Then $Q$ says $s'=s+\Sigma$(remaining terms); in words, the final sum is the sum so far (the sum we have already accumulated) plus the sum of the remaining terms.

Sometimes it's hard or impossible to change $P$ from saying the whole problem into saying the remaining problem somewhere in the middle of execution. So here's another possibility for $Q$. We describe what's been done so far, let's call that $A$, and then define $Q$ as $A\Rightarrow P$. $Q$ says: given that we've done $A$, finish doing $P$. But try the suggestion of the previous paragraph first. When it works, it's a simpler specification than the one in this paragraph.

After we have defined $P$ and $Q$, we have to prove the two refinements. The (initialization) is usually some assignments, so the proof of the $P$ refinement is just some uses of the Substitution Law.

The $Q$ refinement can be proven by cases. The first case is

$(\text{done}) \land \text{ok} \Rightarrow Q$

The other case is

$\neg(\text{done}) \land ((\text{one step}). Q) \Rightarrow Q$

If the (one step) is just some assignments, then $((\text{one step}). Q)$ can be simplified using the Substitution Law.

If $Q$ happens to be an implication, say $A\Rightarrow P$, then the first case is

$(\text{done}) \land \text{ok} \Rightarrow (A\Rightarrow P)$

and that can be changed by portation into

$(\text{done}) \land \text{ok} \land A \Rightarrow P$

The other case is

$\neg(\text{done}) \land ((\text{one step}). A\Rightarrow P) \Rightarrow (A\Rightarrow P)$

which, by portation, is the same as

$A \land \neg(\text{done}) \land ((\text{one step}). A\Rightarrow P) \Rightarrow P$

We might be able to simplify $((\text{one step}). A\Rightarrow P)$ into the form $B\Rightarrow P$. Now we have to prove

$A \land \neg(\text{done}) \land (B\Rightarrow P) \Rightarrow P$
There is $P$ on the left of the main implication that would imply $P$ on the right, but the $P$ on the left has an antecedent $B$ which we need to get rid of. And we can get rid of it if we can prove

$$A \land \neg(\text{done}) \Rightarrow B$$

So that's the proof strategy.