Include sufficient framing for your proofs, so the dots are all connected for the reader. This framing includes stating what you are going to show, any assumptions you make to start your argument, and connecting the dots between the pieces that you've shown and the overall result. For induction proofs:

- Start the whole proof by stating what you are going to show — what is $P(n)$ for your problem?
- Start the base case by stating what you are going to show — what is $P(0)$ for your problem? If the base case is something other than 0, substitute that.
- Start the induction case by stating your assumption ($P(k)$), then state what you are going to show ($P(k+1)$). Write what $P(k)$ and $P(k+1)$ are for your problem — don't just write $P(...)$.
- At the end, conclude that because you've shown $P(0)$ (the base case) and that $P(k) \to P(k+1)$ (the induction case), you've shown $P(n)$ for $n \geq 0$.

Note that you don't need to use the terminology $P(n)$, $P(0)$, $P(k)$, $P(k+1)$ directly — write what those statements are for the specific problem.

Be careful to indicate when you are writing something that you wish to show and something that is assumed, both to distinguish these cases from each other and to distinguish them from deductions.

For #4, be careful not to mix up what you are trying to show (that the function `sum(arr,n)` returns the sum of array elements `arr[0]`, `arr[1]`, ..., `arr[n-1]` for all $n \geq 4$) with the code itself (which is how you will show that `sum` works correctly).

The structure of an induction correctness proof matches the structure of the recursion: the base case for the proof is that the base case of the recursion does the right thing, and the induction case for the proof is that the recursive case in the code does the right thing. The assumption of $P(k)$ to show $P(k+1)$ means assuming that recursive calls with smaller problems work correctly.

Loop invariants provide a way to apply induction to loops: the invariant is something that is true before the first iteration, between successive iterations, and after the last iteration. The base case is to show that the invariant is true before the first iteration and the inductive case is to show that if the invariant is true before one iteration, it is still true after. That the iteration is true after the last iteration is used to show that the entire loop did its job. For #4b, the stated loop invariant is that s is the sum of elements `arr[0..i-1]` — the base case is to show that the invariant holds when $i = 1$ (before the first iteration, so only the statements before the loop have executed) and the inductive case is to show that if the invariant holds when $i = k$ (before iteration k), it still holds when $i = k+1$ (before iteration $k+1$). Use what happens in the loop body to show this. The final step is to show why the loop invariant being true when the loop ends ($i = n$) means that the overall claim that `sum(arr,n)` works correctly is true.

For #8, present the argument more formally, using the definitions and the boolean algebra for sets rules (with justifications) to show the result. (Treat $A \cap B = B$ as the premise and $B \subseteq A$ as the conclusion to deduce.)