How to Prove Decidability or Turing-Recognizability

To prove that a given language is decidable:

- Construct an algorithm that decides the language.
- This algorithm may "call" any other algorithms from the textbook, lectures, class handouts, or homework assignments (but you should cite the appropriate reference).
- How do you decide which existing algorithms it should call, if any? This is a familiar problem for any program designer. You may be expected to figure this out for yourself or a hint may be provided. When proving closure of the class of decidable languages under a given operation the obvious choice is an assumed decider for a given decidable language.
- Prove that the language it recognizes is equal to the given language and that the algorithm halts on all inputs.

To prove that a given language is Turing-recognizable:

- Construct an algorithm that accepts exactly those strings that are in the language. It must either reject or loop on any string not in the language.
- This algorithm may "call" any other algorithms from the textbook, lectures, class handouts, or homework assignments (but you should cite the appropriate reference).
- How do you decide which existing algorithms it should call, if any? This is a familiar problem for any program designer. You may be expected to figure this out for yourself or a hint may be provided. When proving closure of the class of Turing-recognizable languages under a given operation the obvious choice is an assumed recognizer for a given Turing-recognizable language.
- Prove that the language it recognizes is equal to the given language.