Based off George Polya's How to Solve It, additions by Annie Chen, updated Fall 2017 by Amanda Vosburgh

How to Solve a CS50 Pset

1. Understanding the Problem

"*What is the unknown? What is the data? What is the goal? Break the problem into parts*"

What is the spec asking you to do? What types of input will you be taking? What should your program be outputting?

Steps: Write down the requirements for the spec in your own words. Watch the walkthrough.

2. Devising a Plan

"*Have you seen it before? Do you know a related problem? Tackle the problem one part at a time*"

How can you turn the inputs into outputs as a person? How can you get the program to do the same thing? What assumptions/prior knowledge did you as a person use? How can you get your program to know the same thing?

Steps: Using the sample inputs from the spec, figure out (without code) how to get to the outputs. Write down the steps you used to do that. Write down the pseudocode/hints from the walkthrough. Outline the entire program in pseudocode in your source code file.

3. Carrying Out The Plan: Use Examples! Use Google!

Turning concepts into code is often the difficult part, especially if you don't know how to start. What problems have you seen like this before? (In section or lecture?)

Steps: Look back on the source code from the week's lecture/section. How can you change that code to use in your pset? Figure out how the code snippets from the walkthrough fit into your pseudocode. Search reference50 and Google for more information on the functions mentioned in the psets. Understand what inputs they take and what they return. Fit them into your pseudocode.

4. Looking Back: Test Your Code!

"*Can you check the result? Can you derive the result differently?*"

How does your program handle invalid input? Does it ever crash? Could you have implemented the solution more efficiently?

Steps: Run check50. Pinpoint what inputs (if any) your program is failing on, and figure out why they're failing. For help, use CS50 Discourse or Office Hours. Look carefully through your code for areas of improvement. Factor out common code, comment to make intent clear, fix style. When pset is graded, review feedback. Ask for clarification if needed. Watch Postmortem video. Consider how your code could be further improved, and different ways solution could have been implemented.

Checklist

- ❏ **Understand the Problem**
  - ❏ Write down the requirements for the spec in your own words.
- ❏ **Devise A Plan**
  - ❏ Watch the walkthrough.
  - ❏ Write down the pseudocode/hints from the video.
  - ❏ Using the sample inputs from the spec, figure out (without code) how to get to the outputs.
    - ❏ Write down the steps you used to do that.
  - ❏ Outline the entire program in pseudocode in your source code file.
- ❏ **Carry Out the Plan**
  - ❏ Look back on the source code from the week's lecture/section. How can you change that code to use in your pset?
  - ❏ Figure out how the code snippets from the walkthrough fit into your pseudocode.
  - ❏ Search reference50 and Google for more information on the functions mentioned in the psets.
    - ❏ Understand what inputs they take and what they return.
    - ❏ Fit them into your pseudocode.
- ❏ **Check Your Work**
  - ❏ Run check50.
    - ❏ Pinpoint what inputs (if any) your program is failing on, and figure out why they're failing.
    - ❏ For help, use CS50 Discourse or Office Hours.
  - ❏ Look carefully through your code for areas of improvement.
    - ❏ Factor out common code.
    - ❏ Add comments to make intent clear.
    - ❏ Fix style.
  - ❏ When pset is graded, review feedback.
    - ❏ Ask for clarification if needed.
    - ❏ Watch Postmortem video.
      - ❏ Consider how your code could be further improved, and different ways solution could have been implemented.