Brian Leavell
CSE 321

Problem Solving Algorithm

1.) Read the problem and break it down

- After reading the problem, go back and try to highlight the key details mentioned. If there are tasks that seem to go hand-in-hand, mark that down. Make sure you can find a distinction between steps within the problem so that you can find a clear starting point.

2.) Find the "Point"

- Now that the problem has been broken into pieces, try to determine what the main point is. This will make it easier to review your own material or do some research (online/books) to brush up on anything you may have forgotten about.

3.) Hardware, Software, Paper (ware?)

- Try to identify what you are going to need for a step of the assignment or the assignment as a whole. This could range from a notebook to write down logic statements, a breadboard for circuit building or your computer to write a new program. This will further divide up the work and make sure we have one thing to work on at a time. During this time you should also be able to identify and work around any specific constraints given to you, such as not having access to certain hardware or having to remain within a certain budget.

4.) Square One

- Now that the problem has been broken up and you are aware of what resources are needed, get started on solving your first step as identified in step 1 of this algorithm. This "starting point" could mean writing a new program right out the gate or just drawing out diagrams on paper. A good idea is to try and identify the behavior of the problem, commonly done through visuals such as charts or diagrams.

5.) Progression and Testing

- As you progress through the assignment and knock out things on the to-do list, make sure your progress doesn't go unchecked. Especially since a lot of parts work together, it is important to double-check your work and be certain that it works as intended. This prevents a major backup if you reach step 9 of 10 and realize all of the work you did before was wrong or doesn't function. (This is very important when dealing with software and hardware). Also, keep in mind that it is good practice to keep your work organized and labeled since this might cause a lot of headaches if you misplace a wire or lose a document that has only part of the problem on it.

6.) The Finale

- Once you have completed your tasks, debugged and double-checked everything, you can now compile your work (neatly) and analyze it as a whole. If the outcome is not as expected, then go back to step 5 for more debugging.