Vantage 140 APSC 160

Implementation Comparison

Lesson Plan:
Function design with 1D arrays, comparing implementations.

Intended Learning Outcomes:
- Practice tracing loops/nested loops through arrays
- Develop skill for describing an algorithm (the steps the code is following to solve a problem) in plain English
- Develop an understanding of the benefits of good software engineering practices:
  - Good names for variables and functions
  - Documentation
  - Testing
  - Modular decomposition of the problem
- Negotiate/reason about pros/cons of competing solutions to a problem

Required Materials:
1) Week 8 Handout

Lesson Procedure		
	Resource(s)	**Time**
Task 1 – Work through Version Bad	Insertion-Sort-VersionBad	20 mins
Get them to trace through Version-Bad - What is the program doing? - Can you break down the steps that are happening in each iteration of the outer loop? - Write an English description of the steps that this program is going through This will take them some time. Some will get there and many won't		
Main Task 2 – Work through exercise	Insertion-Sort-VersionGood	30 mins
Give them Version-Good to trace through - What is the program doing? - Write an English description of the steps that this program is going through Ask them which program is better/worse? Discussion of pros/cons. A table of discussion points provided on next page.		

Version Bad

- Shorter
- Not a lot of lines of code to read
- Had to trace through entire program to understand what it was doing
- No documentation
- Variable names are not helpful in understanding the program.
- One big piece of code with loops nested in loops makes it hard to trace
-

Version Good

- Longer
- More code to read but…
- Didn't even need to read the code to understand what the program was doing.
- Names of functions, parameters and variables helped in the comprehension of the program.
- Testing the function with multiple inputs and printing the result convinces us the code is correct
- Decomposing the problem into pieces (separate functions)
    - Easier to write the code for those small functions
    - Makes the code easier to understand
    - Makes it easier to test the individual functions if the program is not work as expected.
    - Makes those functions reusable for other parts of your program to call.
- Someone updating this code would more easily find the place to make changes/updates.
-