What you should know prior to COP 3531, File Structures

1. **Requirements**: You should understand that you should never start the design and implementation of a programming assignment until you fully understand what is being requested. To do so wastes your valuable time and effort. If you are uncertain as to 'what' the requirements are, then please ask.

2. **Design**: We will, as is customary, approach design from two perspectives: preliminary design (architectural design) and detail design. For architectural design, your design will consist of structure charts indicating hierarchy and dependencies among program modules. Once your architectural design is completed and verified (appears to meet the requirements), then detail design will be such that each module identified in architectural design will have an accompanying pseudo-code file associated with it. Here, you will outline your algorithmic approach to your problem solution.

You should know about Structure Charts and Pseudo-code (Chapter 5 in your textbook).

3. **Implementation**. **THEN, programming** (implementation) may start. Start with the top modules and use drivers and stubs to SLOWLY implement your design. Verify as you go. Using probes and techniques we will discuss in class, this is the way to proceed.

4. **Test**. Each module should be **tested** as thoroughly as possible. We will call this unit testing and will elaborate much more on this during your class. We will discuss various types of testing – white box and black box testing, verification, validation, etc.

There is a definite discipline to programming that is frequently not conveyed in COBOL. This is generally acceptable, in that students are often learning the notion of programming, syntax, development environment, etc. But as you advance into more meaningful programs, there are smart ways to do things, and …. not so smart ways to do things. Working 'smarter' not 'harder' is the key. Gauge yourself and take the short way to gradual, successful, incremental implementation, verifying as you go.

The keys to success are simple: Don't miss class. Study every day. Jump on the programs immediately – 'front-end' them. Don't get behind, as it will be next to impossible to catch up in a short eight-week summer session.

Work smart and hard and follow my recommendations and you will succeed.

COBOL Topics: The ones you should know or get up to speed as soon as possible.

Format:

- All conditional statements are to have scope terminators on them.
- Indentation is absolutely essential and critical
- Each paragraph is to have a blank line above it
- Only one statement per line
- IFs and ELSEs will align; Perform and End Performs will align, etc. All control statements, such as these will have ending scope terminators which are to be aligned. (same for inner Ifs, nested Performs, etc.) If you have questions, ask.

Input/Output:

- Always Close your files
- ALWAYS use an At End with your Read for sequential file access
- There is to be only one Stop Run in your driver program, and it is to be implemented in a top module. (Equivalently, all PERFORMS are to be 'completed.')
- Understand line counters and page counters (with page number suppression, etc.)
- Understand all about Control Break processing (major and minor control breaks)
- Understand how to format print lines, and Write…From statement for headers, trailers, etc. (but NOT for detailed line) from Working Storage.

MOVES:

- Know the differences between a numeric move and an alphanumeric move. Undertstand group moves.
- Understand which Moves are valid and which are invalid.
- Understand all the particulars in moving short fields to longer fields and vice versa.
- Understand ALL editing symbols and their use.

Arithmetic Statements

- Understand all Add, Subtract, Multiple, and Divide formats
- Understand the Compute and order of operations; parenthesizing; exponentiation.
- Understand On Size Error options and all that is related to this.
- Understand not only how the arithmetic statements operate, but what happens when the result of the operation is Moved to a short/long receiving field.

PERFORM:

- Understand the meaning of the pre-test and post-test as it applies to the Perform.
- Understand and be able to code/implement Performing paragraphs and in-line Performs. For the latter, understand the indentation and nesting possibilities.
- Clearly understand the number of times (if appropriate) a loop will be executed

- Ensure you fully understand the sequence of test, then, execute increment, test sequences for the Performs.
- Be comfortable using the Performs with single-dimensional arrays by using a subscript / index for both control (number of times a loop is iterated) and control (using the loop control variable for accessing an element of an array).
- Understand the differences between subscripts and indexes
- Understand what really happens when subscripts are used to access elements of an array.
- I find many students typically don't really understand how the Perform works. Try to verbally explain how the Perform works….

Tables:

- Understand subscripts and indexes and their differences.
- Understand single-dimensional arrays well.
- Understand Search and Search ALL verbs and HOW to use them.
- Really understand how the indexes function with Search and Search All.
- Understand single-level OCCURS clauses.
- Understand single-level OCCURS clause on a group item and how to access the elementary items below.
- Understand the Redefines
- Understand the Redefines when used with the OCCURS for table processing
- Understand 'implicit redefinition' on the 01 level in the File Section.
- Anything on multiple-level tables will be a plus…

Others:

- Understand the Evaluate
- Understand how to handle alphanumeric data that is 'read in' to numeric fields (think Redefines).
- Understand the PIC 999 fields are really not numeric fields ready for computation.
- Understand Comp-1, Comp-3, Display formats and their differences and efficiencies.
- Understand about data validation (Class tests, Sign tests, etc.)
- Understand what is meant by a logical file and a logical read/logical write.
- Understand what you are looking at using Cobol Net-Express.