Asgn 1 (CS3310 S15) Demo Specs (& Related Notes)

Pseudocode for Main

```
• DELETE 2 output files:  Log.txt, IndexBackup.txt
  (Yes, this should be done by the code, even if you manually deleted these OR you're
  opening them in truncate mode, when appropriate).
• Call setupMain – sending in "AZ" for fileNameSuffix
      (where that method will concatenate it with the pathString, "RawData" & ".csv"
       which are all hard-coded)
• Call prettyPrintMain – no parameters need to be supplied since that method
  automatically uses IndexBackup.txt (hard-coded into code)
• For loop with i going from 1 to 3
      Call userAppMain sending in i for fileNameSuffix
              (where that method will concatenate it with the pathString, "TransData"
               & ".txt" which are all hard-coded)
• Call prettyPrintMain – no parameters needed
```

WHAT TO DO FOR THE DEMO

1. RawDataAZ.csv and the 3 TransData?.txt files must be in correct folder in your project
2. Run the program
3. Print Log.txt file in WordPad or…
    - *Use a FIXED-WIDTH FONT (like Courier New) so things line up nicelyl*
    - *Use a smaller font, if needed, to avoid any wrap-around in Log file printout*
    - *NOTE: This in ONE LONG FILE which includes the output from main running setup, prettyPrint twice and userApp three times – all captured in a SINGLE Log* file
4. Print out IndexBackup.txt file in WordPad or . . .
   (Yes, prettyPrintMain already printed it to the Log file, with things nicely aligned).
   (Yes, things won't line up necessarily, fields may beright next to each other without separators, there may be extra stuff if you used serialization, there may or may not be <CR><LF>'s in the file)
5. Print all of your program code files.

WHAT TO HAND IN (in the order specified below)

1. Cover sheet (fill in the top & sign it)
2. Printout of Log.txt data file
3. Printout of IndexBackup.txt data file
4. Printout of YOUR code files: *(IN THIS ORDER) (There are at least 5 actual separate files]*
    - main (testDriver) program
    - Setup file (including setupMain and any of its private methods, if any)
    - UserApp file (including userAppMain and any of its private methods, if any)
    - PrettyPrint file (including prettyPrintMain and any of its private methods, if any)
    - CodeIndex OOP class file
    - BSTNode OOP class (which may be part of CodeIndex class file or it may be separate)
    - any other code files you used in your program

HOW MUCH COMMENTING IS NEEDED?

- **Self-documenting** code including:
    - descriptive **NAMING** of programs, methods, classes, objects, records, fields, namespaces/packages, variables, constants, etc. *[according to traditional C#/Java/C++ naming conventions]*
    - using the same naming as in the **SPECS** (so everyone's on the same page)
    - good **MODULARIZATION**, short modules (no method > 1 page/screen-ish), sharing of CodeIndex classe and using the modularization described in the specs and in class (so everyone's on the same page)
    - following the **REQUIREMENT SPECS** closely, so that the project designer's specs serve as part of the external documentation (which therefore does NOT need repeating within your code).
- A **top-comment** on each physical file with: overall project/app name, the module name the code author's name & date completed
- Comments on **tricky code** or unusual ways of doing things or things which don't quite follow the specs (since a maintenance programmer would read the specs and ASSUME that the program would OF COURSE follow them)
- You do **NOT need line-by-line** commenting

NOTES:

- Re-read specs for A1 to make sure you're doing everything right (to maximize points)
- Setup, UserApp and PrettyPrint modules all use the input stream processing algorithm (on RawData, TransData and IndexBackup files, respectively) – i.e., loop through the data til EOF, doing 1) read in a single record/line then 2) deal with it.