Asgn 1 Demo Specs (& Related Notes)

WHAT TO DO FOR THE DEMO

1. [RawDataTester.csv & TransData.txt files must be in the correct folder in your project]
2. DELETE Log.txt and NameIndexBackup.txt files
3. Run SetupProgram
4. Run PrettyPrintUtility
5. Run UserApp
6. Run PrettyPrintUtility
7. Print Log.txt file in NOTEPAD (or WordPad or…).
   - *Use a FIXED-WIDTH FONT (like Courier New) so record fields line up nicely for PrettyPrintUtility's output.*
   - *NOTE: This includes the results from steps 3 & 4 & 5 & 6 above (since file is opened in APPEND mode by PrettyPrintUtility and when UserApp uses NameIndex class methods)*
8. Print NameIndexBackup.txt file in NOTEPAD (or WordPad or…).
   - *NOTE: This includes the results from JUST STEP 5 above (since file is always opened in TRUNCATE mode)*
9. Print all of your program code files.
10. Circle what I've described below (by hand).

WHAT TO HAND IN (**in the order specified below**)

1. Cover sheet (fill in the top & sign it)
2. Printout of Log.txt file – CIRCLE THE FOLLOWING:
   - N and RootPtr in PrettyPrintUtility's output (both runs)
3. Printout of NameIndexBackup.txt– CIRCLE THE FOLLOWING:
   - N and RootPtr
   - The newly inserted countries resulting from UserApp's IN TransData handling (Argentina, Japan, Jamaica, Cuba, Austria, Azerbaijan, Botswana, United Kingdom, Canada, Kazakstan, United States Minor Outlying Islands)
4. The 2 BST worksheets (done by hand)
5. YOUR program code: *(IN THIS ORDER) (There are at least 6 actual separate files]*
   SetupProgram program
   UserApp program
   PrettyPrintUtility program
   RawData class
   UserInterface class
   NameIndex class
   any other code files/classes you used in your program
6. CIRCLE THE FOLLOWING in your program code:
   - The increment in BST SEARCH (so I know you're not doing linear search)
   - Any mention of "BST" in SetupProgram or UserApp (so I can take points off)
   - DeleteCountry (or similar name method header inside NameIndex class

##

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 naming conventions]*
  - using the same naming as used in the **SPECS** (so "everyone's on the same page")
  - good **MODULARIZATION** (using OOP for SetupProgram and UserApp, short modules, sharing of the UserInterface & NameIndex classes) 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 your "boss's" specs act as a form of external documentation (which does NOT need repeating within your program).
- A **top-comment** on each physical file with: the module name & the code author's name & the overall app name
- A **comment-line-of-*'s** between chunks of code (e.g., methods, constructor, …)
- Comments on **tricky code** or unusual ways of doing things or things which don't 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)