FOR GRADER FOR ASGN 1

Things to check regarding PROGRAMMING points

- Appropriate **commenting** on each file
- Project has these **MODULES** with appropriate names (matching specs)
  - Setup & UserApp are PROGRAMS with their own main's (plus there's a BORROWED PrettyPrint, not their own version, except for Jia & Devlin who wrote the shared-with-the-class versions)
  - 2 shared OOP classes, DataStorage & UIoutput, (same exact 2 classes/files used by BOTH Setup & UserApp – not duplicate copies used by each program)
  - 2 non-shared OOP classes: RawData (ONLY used by Setup) & UIinput (ONLY used by UserApp) – the "other" program does NOT declare an object it shouldn't own and just not use it
  - There are physically separate code files for each of these programs/classes
- **COMMUNICATION** structure
  - RawData & DataStorage do not talk to each other – they can only communicate via Setup program (i.e., which calls methods in those 2 classes)
  - Same for UIinput & DataStorage communicating only through UserApp
  - Both Setup & UserApp pass the uiOutput object in to appropriate methods in RawData & DataStorage & UIInput classes, so that those methods can call uiOutput.displayThis method (for writing to Log file)
- Setup & UserApp are **CONTROLLERS** who don't do much actual work except
  - declare 3 objects at the top
  - call finishUp for those 3 objects at the end
  - call uiOutput.displayThis to write "program starting" and "program stopping" status messages at the top(ish)/bottom(ish) of their main
  - [these programs DO NO write status messages for opening/closing any files, nor call uiOutput.displayThis to write such messages - that's handled by the OOP classes in charge of the files]
  - Each program has a loop (til done) containing
    - Call rawData.input1Country or uiInput.input1Trans
    - Call dataStorage.insert/delete/select/showAll (or some similar name) to handle things
  - UserApp's loop contains a big switch statement (based on transCode) to handle cases for I, D, S, A, %
- RawData & UIinput classes do **NOT contain**:
  - a big array for storing the entire file – it only has storage for a SINGLE RECORD/LINE (plus other variables for splitting, etc., as needed)
  - Looping code to control reading in multiple records (though it could have a loop for reading field-by-field) – that's Setup's/UserApp's responsibility
- RawData & UIinput & UIouput classes' each **deal with their own file**
  - constructors OPEN their respective file
  - finishUp methods CLOSE their respective file
  - contain status messages "requests" in their constructor & finishUp methods (i.e., calls to uiOuput.displayThis)
- **DataStorage** class contains
  - EITHER an array of bstNode objects OR 9 parallel arrays
  - 4 Separate public methods for insert/delete/select/showAll (or similar names)
  - A call in the constructor to loadBstFromFile method (or similar name)
  - A call in finishUp method to saveBstToFile method (or similar name)
  - showAll method uses inOrderTraversal algorithm and NOT A SORT
  - select/delete uses a bstSearch algorithm and NOT A LINEAR SEARCH
- **Backup** file may be handled EITHER within DataStorage class OR in a separate class (which is only known to DataStorage, and not to Setup or UserApp)
  - loadBstFromFile method handles opening/processing/closing file & appropriate status message handling
  - saveBstToFIle method handles opening/processing/closing file & appropriate status message handling

Things to check regarding OUTPUT points

- Uses a fixed-width font (like Courier New) so things align
- Uses landscape and/or a small enough font so there's NO WRAP-AROUND
- Programs run in the correct order (Setup, PrettyPrint,UserApp, PrettyPrint)
- Log file contains the cumulative results from running Setup, PrettyPrint, UserApp, PrettyPrint (and not the accumulated results from prior testing)
- Printout of Backup file's in the demo packet

- Status messages appear at the correct time (reflecting a “trace” of what’s happening when the programs execute)
- Log file does NOT open/close/open/close/open/close/… during the run of a single program - it only opens ONCE & closes ONCE during Setup’s run, and similarly for UserApp & PrettyPrint
- OUTPUT IS CORRECT
- ATA (Antarctica) is included (and not lost, just because it’s the last record in RawData file
- Output is correctly formatted (You can forgive an extra space or blank line here or there, but developer must demonstrate a concerted effort to follow the specs in terms of format)
- # nodes visited is reported for every transaction request (except A transactions) and is correct for both successful and unsuccessful requests
- LeftChPtr and RightChPtr are correct (in PrettyPrint’s results)
    - After running Setup
    - After running UserApp (which contains inserts/delets)
- Searches for deleted (tombstoned) countries and D requests for deleted (tombstoned) countries results in “invalid country code” messages
- Deleted countries do NOT appear in A (ShowAll) transaction requests’ results