Assignment 3

(Due Friday of week 8 - May 10)

1. Write programs for the following exercises in Java, Python, or C#. Each file should have your name at the top in comment, with short description of what that file is implementing. Make sure your files have appropriate names. Programs should write output to the Console and have input hardcoded in main.

   Note 1: If a program is not in approved programming language (or in different language than previous assignment) or has any syntax error, no points will be awarded for that exercise

   Note 2: Submitting wrong files or in the wrong format or corrupted files will not be accepted nor will any re-submission be allowed for any such mistake. It is your responsibility to submit all the files in the correct format on time.

   a) **QuickSort:**
      1. Implement the pseudo code we went over in class for Quicksort. Make sure to use the same method names and variables. Write test code passing the following elements [43, 1, 32, 7, 12, 56, 2, 14] to the sort and print the values before and after being sorted to the console. In the video explain the quicksort algorithm, run the test program and discuss the output.
      2. Modify the quicksort to use as pivot a) first element, b) last element, and c) random element.
      3. Capture runtime (how long it took to sort) for 3 different data sets: random values, already sorted, and sorted in opposite order of at least 500-1000 elements each for each version of the quicksort.
      4. Write a report discussing what had to be changed for each pivot version, how you generated data sets, runtime results of all 4 quicksort versions against the 3 data sets, how the results compare, and your conclusions about the algorithms based on your results and what you learned about the quicksort.
      5. Submit all quicksort code versions in one zip/rar file; analysis report as PDF or Word document to be checked by TurnItIn;

   b) **BST:**
      1. Implement Binary Search Tree ADT (call the class BST) for the pseudo code we went over in class. Make sure to use the same method names and variables. You need to have all the operations that the pseudo code has. You should have a single file with class BST for the program. Your test program (main method) should create an instance of BST and demonstrate that all operations work correctly. In the video explain implementation for each operation, run the test program and discuss the output.
      2. Make a copy of the above BST code calling it BST2. Add to Node another attribute called status of data type boolean (you can change Node to Node2 if you want to keep all code in single folder). Then add a new BST operation called deleteFalse with no

parameters that deletes all nodes in the tree which have status value false. Write test program to demonstrate that the new operation works correctly. In the video explain implementation of the deleteFalse operation, run the test program and discuss the output. Analyze the time and space complexity of the new operation.

3. Submit all code in single zip/rar file

2. Record a video 10-12min discussing each exercise as indicated above.

Grading Rubric

Points	**Criteria**
10	Programs use object oriented program approach, have the appropriate naming convention as indicated in the exercise, author's name, and brief description of the problem/implementation in each file
45	**QuickSort:** Correctly implements Quicksort pseudo code that was given in class with the same naming of methods and variables Correctly modifies code for different pivots: first element, last element, random element Analysis is submitted as PDF or Word file Correctly generates 3 data sets for the analysis and describes in the report Correctly captures runtime for all algorithm versions and data sets Correctly analyzes the runtime performance of the sort, provides results and conclusions in the report Video explains implementation of the Quicksort, runs for the given test data, and explains output Video briefly explains test data used, analysis, and results of the sort performance
45	**BST:** Correctly implements Binary Search Tree ADT for all the operations in the covered pseudo code with the same naming of methods and variables Correctly copies BST to BST2, modifies Node, and adds deleteFalse operation Test program for BST demonstrates correct execution of all operations Test program for BST2 demonstrates correct execution of new operation Includes time/space analysis of the deleteFalse method Video explains implementation of BST ADT, test program, shows program running, and explains output Video explains modifications and new operation in BST2 ADT, test program, shows program running, and explains output