Assignment 3

(Due Friday of week 8 - Dec 21)

1. Write programs for the following exercises in Java, Python, C#, 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.

   a) **MergeSort:** Implement data as raw array and implement merge sort (to sort in ascending order) that we discussed in class (recursive version) to sort the data. Compare the runtime performance of the sort using different test data (i.e. small set, large set, sorted set, partially sorted, sorted in opposite order, random) and discuss how your results support what you have learned about this algorithm in class. Write up the details of what test data you used, analysis of the execution, and your conclusions. Your program should be implemented as a single file with class MergeSort, method mergesort (and any others you may need for the algorithm), and main method to populate array, call mergesort method to sort, and display sorted array. For output show the result of the following input: [2, 98, 3, 14, 5, 65, 1]

   b) **BST:** Implement Binary Search Tree ADT for the pseudo code we went over in class. Make sure you 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.

   c) **SumAverage:** (1) Add a new operation to the above BST class called sumNodes which will traverse the tree, sum up and return node values. (2) Add new operation to the above BST class called averageTree which calculates the average of the nodes values. Evaluate and report the time and space complexity of your sumNodes and averageTree algorithms. Your test program (main method in separate test class and file) should create an instance of BST, populate it with below values, and demonstrate that both new operations work correctly.

      For example,
      Given a BST with the following values: 90 21 38 50 40 45 87
      sumNodes should return: 371
      averageTree should return: 53

If the tree is empty or sum is 0, both operations should return 0

2. Record a video 10-12min long explaining the implementation, solution, and output of each of the above programs.

3. **Submission instructions:** Submit one zip file with all the programs, second file with sort performance analysis for program and SumAverage exercise time/space complexity analysis, and third zip file with the video (if you have multiple videos they each should be a separate zip file)

Grading Rubric

Points	Criteria
10	Programs use object oriented program approach, have the appropriate naming convention, author's name, and brief description of the implementation in the files and the problem it is solving for main() methods
30	**MergeSort:** Correctly implements raw array for data Correctly implements merge sort algorithm that was covered in class Correctly selects test data for the analysis and describes in the report Correctly analyzes the runtime performance of the sort, provides results and conclusions in the report Video explains implementation of merge sort using raw array Video explains test data used, analysis, and results of sort performance Video shows the running program for given instance problem and explains output **For extra credit:** manually show the array contents after each iteration of the sort and explain what it is doing
30	**BST:** Correctly implements Binary Search Tree ADT for all teh operations in teh covered pseudo code. Video explains implementation of BST ADT, test program, shows program running, and explains output
30	**SumAverage:** Correctly implements and adds sumNodes and averageTree methods to BST ADT Test program creates instance of BST and loads with given data Test program calls the new methods and prints the result There is correct time and space analysis of new operations using big O notation and video explains it Video explains implementation and solution of new operations Video shows and explains running program for example data and explains output