Step 1: Implementation [30 points total, 5 points for each function of a data structure]

Perform an analysis of *insertion, traversal, and deletion* of AVL Trees and another non-linear data structure. You are free to choose the other non-linear data structure e.g. Binary Search Tree, Min Heap, Max Heap, B+ Tree, Splay Tree, Red Black Tree, or something of interest to you outside the ones covered in this course. You have to implement these functions on your own and not using pre-built functions in libraries for the exact same data structure. This means you can use lists or arraylists, but not priority queues from STL if implementing heaps.

Step 2: Generation of Input [9 points total, 1 point each file]

Next, you are supposed to perform this analysis on nine input files consisting of generated numbers. The files have two properties: size and input order. The size consists of three categories: small, medium, and large. Input order also consists of three categories: ascending, descending, and random order. Make sure that the three sizes have a difference of at least a factor of 10 and the smallest file size has at least 100 numbers. **For example, you can have three sizes as 500, 5000, 50000 or 1000, 10000, 100000.** However, these sizes 500, 5000, 10000 are incorrect as the the difference between 5000 and 10000 is not a factor of 10. This will yield nine files i.e., Small Ascending, Small Descending, Small Random, ..., Large Random.

Step 3: Application of Input to Implementation and Recording Time [27 points total, 0.5 points for each time entry]

Perform each of the functions, *insertion, traversal, and deletion* on the numbers in the file and record the time taken for each function. For example, for the small ascending file of size 100, you will start your clock for an empty tree, insert all 100 numbers one by one and stop clock after all inserts. This will be time $t_{sm\_as\_in}$. Next, you will start your clock for the tree with 100 nodes, traverse all 100 numbers one by one and stop clock after all numbers are printed. This is time $t_{sm\_as\_tr}$. Finally, you will open the small ascending file again, start the time, and start deleting each number in the file from your tree one by one until the tree is empty. This is time $t_{sm\_as\_de}$. You will repeat the process for the remaining eight files. This will give you 27 time entries. Make sure you have these time entries in a table. Repeat this process for another data structure. Thus, you will have a total of 54 time entries which should be present in your report with the appropriate time units.

Example Table for Insert of Data Structure 1:

	Ascending	Descending	Random
Small	$t_{sm\_as\_in}$	$t_{sm\_ds\_in}$	$t_{sm\_ra\_in}$
Medium	$t_{me\_as\_in}$	$t_{me\_ds\_in}$	$t_{me\_ra\_in}$
Large	$t_{la\_as\_in}$	$t_{la\_ds\_in}$	$t_{la\_ra\_in}$

Step 4: Practical Commentary and Reflection [22 points total]

Finally, compare the run times of similar functions of two data structures. Example, compare insert function of AVL Trees with insert function of Data Structure 2 and so on for traversal and deletion functions. Also, comment on the file size as well as the order of the input and its relationship to time complexity. This commentary is worth 10 points. The remaining 12 points is for plotting graphs that show the run-time of the times you noted in Step 3. These graphs will also be graded based on aesthetics so make sure you display the results so that it is intuitive for a novice person to pick one data structure over another in a given scenario by seeing your graphs.

Step 5: Theoretical Commentary and Reflection [6 points total]

Comment and describe the theoretical computational complexity in terms of Big O for the three functions of both your implemented data structures.

Step 6: Learning Commentary and Reflection [6 points total, 3 points each]

Comment on what you learned and what challenges you encountered for this project.

Your final submission should include:

1. Your implementation of both data structures either in one file or in two files in the respective language format, .py, .cpp, .js, .c, .cs, or.java. **Thus, Java, C#, C, C++, Python and JavaScript are the only allowed languages.**

2. Three input files of largest size in .txt format.

2. Your full report consisting of Step 3, 4, 5, and 6 in a .docx, .doc, or .pdf format.

- Note that only the above types of file uploads will be allowed and you cannot submit .rar files or .zip files, or another type of files.