LEBANESE AMERICAN UNIVERSITY
School of Arts and Science
Department of Computer Science and Mathematics
CSC 310: Algorithms and Data Structures
Lab 2

Implement the class **AVLNode** which represents an AVL tree node having an integer value, integer height and references to the left child and right child, as well as a constructor that takes an **integer** as argument. Using **AVLNode**, implement the class **AVL** representing an AVL tree.

In the **AVL** class, implement the insert method, which takes as input an integer value and adds it to the tree maintaining the AVL property of the tree by doing the right rotations (for every node, require heights of left & right children to differ by at most +1 or -1).

Problem 1 - *AVL insert & sort:*

Given a sequence of integers, insert them into the constructed **AVL** tree maintaining the AVL property of the tree by using rotations (for every node, require heights of left & right children to differ by at most +1 or -1), then print the constructed AVL tree using **in-order** traversal.

The first line of input is an integer **T** representing the number of test cases. Each test case is made up of an integer **N** representing the number of nodes in the tree followed by **N** integers representing the values to insert.

Sample Input	Sample Output
3	
7 25 13 10 30 8 27 37	8 10 13 25 27 30 37
4 5 6 8 7 4	4 5 6 7 8
6 10 7 15 13 4 6	4 6 7 10 13 15

Problem 2 – *Merge Sort:*

Given an array of integers, write a program that sorts the array using ***Merge Sort***.

Sample Input	Sample Output
7	
12 700 9 156 34 -732 237	**-**732 9 12 34 156 237 700

Problem 3 – *Quick Sort*:

Given an array of integers, write a program that sorts the array using the ***Quick Sort*** algorithm covered in class (pivot is first element of array/subarray).

Sample Input	Sample Output
7	
12 700 9 156 34 -732 237	-732 9 12 34 156 237 700

Problem 4 – *Heap Sort:*

Given an array of integers, write a program that sorts the array using the ***Heap Sort*** algorithm using max heap. You should implement the MaxHeapify and sort in a **MaxHeap** class.

Sample Input	Sample Output
6	
12 11 13 5 6 7	5 6 7 11 12 13

Problem 5 - *Hybrid Merge/Insertion Sort:*

Write a program that sorts the array using ***Merge Sort***. But whenever the length of the (sub)array becomes less than a user-defined threshold (part of the input), use ***Insertion Sort*** to sort the (sub)array.

Sample Input	Sample Output
9 5	
12 700 9 156 34 -732 237 1 2	-732 1 2 9 12 34 156 237 700

Problem 6 – *Homework:*

This is a homework assignment. You are asked to run a race among four different sorting algorithms: the above three sorting algorithms (Merge Sort, Quick Sort, heap sort, and insertion sort). Test which one is faster on different (very large) array sizes, and submit a report by e-mail about the results of your race, and the randomly generated lists and the code by **Friday, 12:00 PM**.
In your report, the final race output should be plotted graphically (plot the average time per test size per algorithm).

For the race, generate random lists of integers (open range for the values) of size **100 to 10^8**, as follows:

100 lists of size 100
100 lists of size 1000
1000 lists of size 10000
Etc…