Design and Analysis of Algorithms Practice (Object Oriented Algorithm Design and Analysis Practice)

Instructions:
1. All assignments are group based, group of size 3. You may choose a group of your choice. There should be equal contribution from each member, so choose your group wisely.
2. Mid-sem and End-sem are individual.
3. You are free to use Internet to refer to any online resources.
3. You are permitted to re-use your code during examinations which you will be developing as part of assignments. Therefore, organize your work and complete your tasks assigned on time. Maintain a folder, use internal documentation (comments for each function)
5. You should maintain a notebook which must contain (i) pseudo code/C ++ code (ii) Trace of your code for sample inputs – for each assignment, you must complete this task before you attempt coding. It is a good practice to write algorithm, trace before implementing. 6. Let your focus be on learning and discovery.
7. GMeet evaluation will be done by TAs. In addition to this, you should email the code of each assignment after evaluation at `assignment.iiitdm@gmail.com`

1 Assignment 1, Due: 9/Aug

Problem: `Finding MIN, MAX, Second MIN, Second MAX in an integer array`
Present 3 different logic. NO logic should use sorting. For example, one of the logic is to divide the array into two equal halves, find min/max/smin/smax for each, recursively update to get the final min/max/smin/smax. Implementation using C++ classes and objects.
Some practice questions, not for submission: Given a number, list all its prime factors, Given m, n, find $GCD(m,n), LCM(m,n)$ (you may think of more than one logic for each).

2 Assignment 2, Due: 23/Aug

Some interesting questions from geeksforgeeks. You may look at the logic if required, however, you must code using C++.
1. Trapping Rain Water:
https://practice.geeksforgeeks.org/problems/trapping-rain-water-1587115621/1
2. Implement two stacks in an array
https://practice.geeksforgeeks.org/problems/implement-two-stacks-in-an-array/1
3. Triplet Sum in Array
https://practice.geeksforgeeks.org/problems/triplet-sum-in-array-1587115621/1
4. Given an array write an algorithm to find the sum of all the possible sub-arrays.
Example: for array = $\{1,2,3,4\}$, All sub arrays are:
[1], [12], [123], [1234],
[2], [23], [234]
[3], [34]
[4]

3 Assginment 3, Due: Sep 6

Divide and Conquer Paradigm
1. Find MAX using 1-way, 2-way, 3-way, k-way approach
2. Count the number of negative numbers in an integer array using 2-way and 3-way approach.

3. Implement an Iterative and a Recursive algorithm for binary search, ternary search.

4 Buffer week: Sep 13

Pending assignments must be eval by this date, and no more eval for A1,A2,A3 after this week.
Mid-Sem: Sep 27, Monday, 14.00 - 17.00. Individual coding and evaluation.

5 Assginment 4, Due: Oct 4

1. Implement Classical Bubble, Selection, Insertion Sort, Merge Sort, Heap Sort.
2. Insertion Variant: Note that at iteration i, the first $(i-1)$ elements are sorted and as part of the insertion sort algorithm, we look for the right position for $A[i]$. The position for $A[i]$ is determined by performing a linear search on $A[1..(i-1)]$. You are asked to implement the following; since $A[1..(i-1)]$ is sorted, perform binary search to identify the right position for $A[i]$. At each iteration i, perform binary search instead of linear search to identify the right position for $A[i]$ so that at the end of the iteration $A[1..i]$ is sorted.

6 Assginment 5, Due: Oct 11

1. Hybrid Sorting: Insertion sort + Merge sort - Consider a large sized array (size at least 1000). Perform Merge sort recursively until the problem size becomes less than or equal to 20. When problem size becomes less than or equal to 20, perform Insertion sort.
2. Implement Quicksort with two pivots; suppose the pivot values are x and y such that $x \leq y$, then Partition 1 has values $< x$, Partition 2 has values between x and y, and Partition 3 has values $> y$.
3. Given a set of m words (strings), arrange them in dictionary order.

7 Assginment 6, Due: Oct 25

1. Implement Greedy Algorithms for Coin Change and Fractional Knapsack.
2. Implement Greedy Strategy for Matrix Chain Multiplication and Optimal Binary Search tree

8 Assginment 7, Due: Nov 1

1. Implement dynamic programming for Coin Change and 0-1 Knapsack
2. Implement dynamic programming for Matrix Chain Multiplication and Optimal Binary Search tree
3. Problems involving polymorphism - check attachment.

9 Assginment 8, Due: Nov 8

1. Implement DFS,BFS, Test for Bipartiteness, Cycle testing and test for connectedness.