ASSIGNMENT - 1
Analysis and Design of Algorithms - CSE222

Question 1: Order the following in increasing order of O(.) notation: **(10 points)**

$n^{\log n}$, $n^{1/2 \log n}$, $2^{\sqrt{\log n}}$, $(\log n)^{\log n}$, n^{10}, 10^6, $(1.0000001)^{\sqrt{n}}$, $(\log \log n)^{20}$

Question 2: The following part contains 'Strange' and 'Abnormal' Sorting Algorithms which use some recursive calls and sort the input. **(25 points)**

- **Strange(A[1 .. n]):**

```
if n > 1
then
        Strange(A[1 .. n/2])
        Strange(A[n/2 + 1 .. n])
        Abnormal(A[1 .. n]
```

- **Abnormal (A[1 .. n]):**

```
if n = 2
then
        if A[1] > A[2]                          // the only comparison!
                swap A[1] ↔ A[2]
        else
                for i ← 1 to n/4                // Swap 2nd and 3rd Quarters
                        swap A[i + n/4] ↔ A[i + n/2]
                Abnormal(A[1 .. n/2])           // Recurse on Left half
                Abnormal(A[n/2 + 1 .. n])       // Recurse on right half
                Abnormal(A[n/4 + 1 .. 3n/4])    // recurse on middle half
```

The comparisons performed by this algorithm do not depend at all on the values in the input array. Assume for this problem that the input size n is always a power of 2.

(a) "**Strange** sorts any input array appropriately." Prove the Statement by Induction. [**Hint:** Consider an array that contains n/4 1s, n/4 2s, n/4 3s, and n/4 4s.]
(b) "If we removed the for-loop from Abnormal then Strange would not correctly sort." Prove the Statement.
(c) "If we swapped the last two lines of Abnormal then Strange would not correctly sort." Prove the Statement.
(d) What is the running time of Abnormal? Give Justification. State the Recurrence Relation.
(e) What is the running time of Strange? Give Justification. State the Recurrence Relation.

Question 3: Solve the following Recurrence Relations: **(20 points)**

1. $T(n) = T(\sqrt{n}) + n$
2. $T(n) = 2T(n/2) + O(n/\log n)$
3. $T(n) = 2^n T(n/2) + n^n$
4. $T(n) = 2 * T(\sqrt{n}) + \log n$ and $T(1) = 1$
5. $T(n) = 8T(n/2) + n^2$ (Solve using **Recursion-tree Method** only!)

Question 4: Let us have some fun with the **Tower of Hanoi** that has been discussed in class. Suppose that the pegs in Tower of Hanoi are numbered **0, 1, 2.** Design an algorithm for solving the Tower of Hanoi problem with a restriction that all the moves were taken to solve the problem **must involve peg 0** i.e no move is possible from peg 1 to peg 2. **(15 points)**

Question 5: Describe and analyze the algorithm to count the number of inversions in the given array A. Time complexity of your algorithm should be O(nlogn). (Mention Space Complexity.)
Note: Inversion in array A[1..n] is if (i < j) and (A[j] > A[i]) then the pair (i, j) is called inversion pair in the array. **(10 points)**

Question 6: Given an array A of integers and an integer k, write an algorithm that returns **True** if the given array can be partitioned into pairs such that sum of every pair is divisible by k, otherwise returns **False.**
Notes: Array contains positive integers only. The program should run in O(n) time complexity **(10 points)**

Question 7: Given an array of n sorted elements, design an **optimal** algorithm for finding if an element 'X' whose index is also 'X' is present in the array or not. (Write pseudo code and Analyze Space and Time Complexity) **(10 points)**