1. Show the steps insertion sort uses to sort the following list of integers in the descending order (from the highest to the lowest / biggest to the smallest):

   $$9, 19, 14, 29, 13, 4, 23, 10, 31$$

   Show the value of the key variable, k, at each step. Explain briefly why time complexity of insertion sort is $O(n^2)$. Use Loop invariant to show its correctness. [5 Points]

Solution.

[9, 19, 14, 29, 13, 4, 23, 10, 37] // k = 19 remains at its position
[19, 9, 14, 29, 13, 4, 23, 10, 37] // k = 14
[19, 14, 9, 29, 13, 4, 23,10, 37] // k = 29
[29, 19, 14, 9, 13, 4,23,10, 37] // k =13
[29, 19, 14, 13, 9, 4,23, 10, 37] // k = 4, 4 remains at its position
[29, 19, 14, 13, 9, 4, 23,10, 37] // k = 23
[29, 23, 19, 14 , 13, 9, 4, 10 ,37] //k = 10
[29, 23, 19, 14, 13, 10, 9 , 4 , 37]
[37 , 29, 23, 19, 14, 13 ,10 , 9, 4] done.

Complexity of Insertion Sort:
$T(n) = c1n + c2(n\ 1) + c4(n\ 1) + c5\sum_{j=2}^{n}(tj) + c6\sum_{j=2}^{n}(tj-1) + c7\sum_{j=2}^{n}(tj-1) + c8(n-1)$
$T(n) = c_1n + c_2(n-1) + c_4(n-1) + c5(1/2n(n+1)-1) + c6(1/2n(n-1)) + c6(1/2n(n-1)) + c8(n-1)$
$T(n) = O(n^2)$
To prove insertion sort is correct, we will use "loop invariants." The loop invariant we'll use is:
Lemma: At the start of each iteration of the for loop, the subarray $A[1 \cdots j-1]$ consists of the elements originally in $A = [1 \cdots j-1]$, but in sorted order.
Initialization: Before the first iteration (which is when $j = 2$, the subarray $A[1 \cdots j-1]$ is just the first element of the array, A[1]. This subarray is sorted, and consists of the elements that were originally in $A = [1 \cdots 1]$.
Maintenance: Suppose $A[1 \cdots j-1]$ is sorted. Informally, the body of the for loop works by moving $A[j-1]$, $A[j-2]$, $A[j-3]$ and so on by one position to the right until it finds the proper position for A[j]. The subarray $A[1 \cdots j-1]$ then consists of the elements originally in $A[1 \cdots j-1]$ but in sorted order. Incrementing j for the next iteration of the for loop then preserves the loop invariant.
Termination: The condition causing the for loop to terminate is that $j > n$. Because each loop iteration increases j by 11, we must have $j = n+1$ at that time. By the initialization and maintenance steps, we have shown that the subarray $A[1 \cdots j-1+1-1] = A[1 \cdots n]$ consists of the elements originally in $A[1 \cdots n]$, but in sorted order.

2.Show the steps merge sort uses to sort the following list of integers in the descending order (from the highest to the lowest / biggest to the smallest): [5 points]

6, 16, 12, 27, 9, 1, 18, 5, 31

Consider the following variation on Merge Sort, that instead of dividing input in half at each step of Merge Sort, you divide into three part, sort each part, and finally combine all of them using a three-way merge subroutine. What is the overall asymptotic running time of this algorithm? (Hint: Use Master Theorem) [5 points]

Solution. [9, 19, 14, 29, 13, 4, 23, 10, 37]

9, 19, 14, 29,13 4, 23,10,37

9, 19, 14 29, 13 4, 23 10, 37

9, 19 14 29, 13 4, 23 10, 37

9 19 14 29 13 4 23 10 37

19, 9 14 29, 13 23, 4 37, 10

19, 14, 9 29, 13 4, 23 10, 37

29, 19, 14,13, 9 37, 23,10, 4

37, 29, 23, 19, 14, 13, 10, 9,4

Time Complexity using Master Method

$T(n) = aT(n/b) + f(n)$

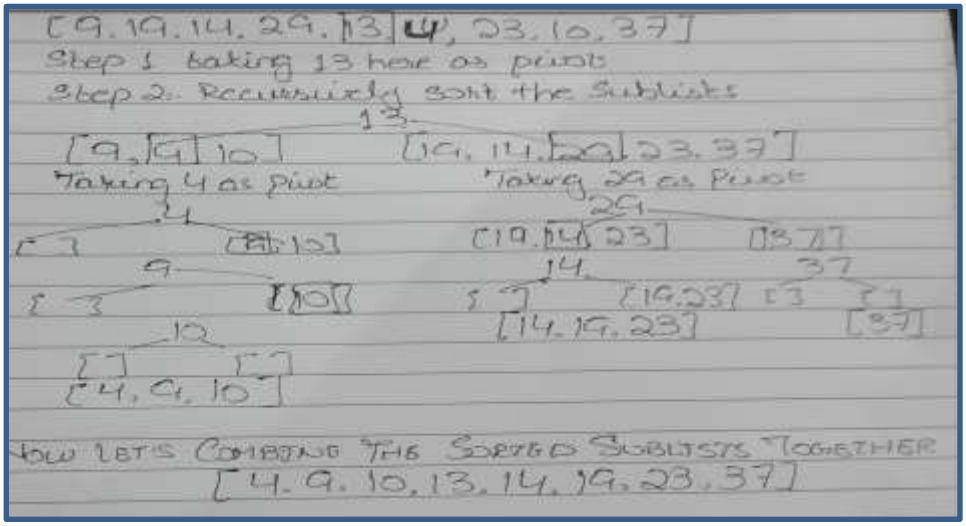
$a = 3, b = 3, f(n) = O(n)$

$T(n) = 3T(n/3) + O(n)$

$T(n) = O(n \log_3 n)$

3.Repeat for Quick Sort. Use Loop invariant to show its correctness. [5 points]

9, 19, 14, 29, 13, 4, 23, 10, 37



Initialization: Before the first iteration, $i = p - 1$ and $j = p$ Because no values lie between p and i and no values lie between $i + 1$ and $j - 1$ the first two conditions of the loop invariant are trivially satisfied. See page no. 173 for further details.

4. Let suppose you are given the task of creating two teams each of n players from the pool of 2n players for the competition. Each player has a numerical rating assigned to him/her according to their talent. Show plausible strategy for dividing players as fairly as possible to avoid talent imbalance between team 1 and team 2.
   Design (i) $O(n^2)$ algorithm to solve this problem [5 points] (ii) $O(n\log_2 n)$ algorithm to solve this problem [5 points]

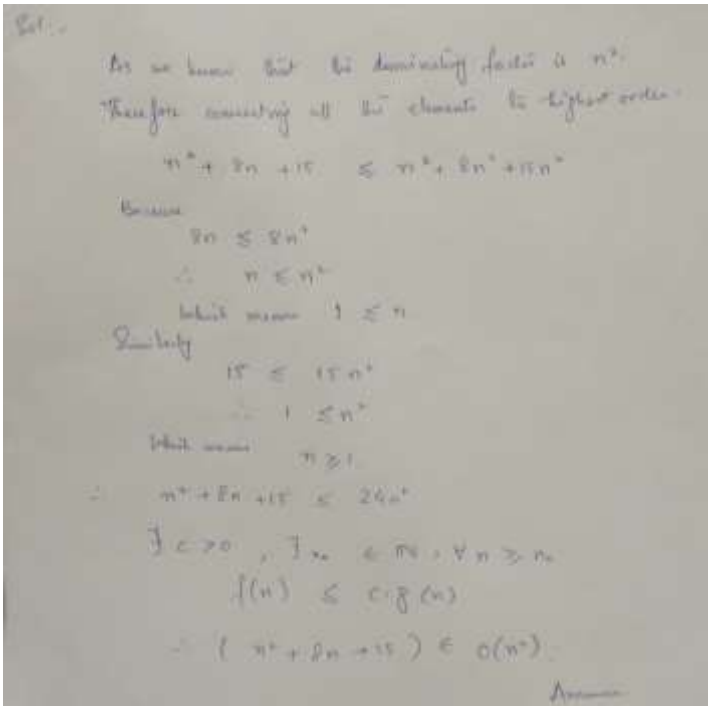
   Solution. Use Sorting Algorithm like Merge Sort to sort players for O(nlogn) and insertion sort for O(n^2). Then select team 1 from odd index (1, 3, 5, 7, ..., n-1) and select other team 2 from even index (2, 4, 6, 8, ..., n) of sorted array.

5. Take a sequence of 2n real numbers as input. Design an $O(n \log_2 n)$ algorithm that partitions the numbers into n pairs, with the property that the partition minimizes the maximum sum of a pair. For example, say we are given the numbers (1,3,5,9). The possible partitions are ((1,3), (5,9)), ((1,5), (3,9)), and ((1,9), (3,5)). The pair sums for these partitions are (4,14), (6,12), and (10,8). Thus the third partition has 10 as its maximum sum, which is the minimum over the three partitions. [10 points]

   Solution. Step 1: Use Sorting Algorithm like Merge Sort to sort. Step 2: Create pairs from endpoints
   x = sort(x) for in in range 1 to n: index1 = i index2 = n - (i+1) new Pair(x[index1], x[index2])

6. Prove $n^3 - 2n + 1 = O(n^3)$. Determine the values of constant c and n_0. [5 Points]
   Prove $5n^2 \log_2 n + 2n^2 = O(n^2 \log_2 n)$. Determine the values of constant c and n_0. [5 Points]

   Solution.



   Prove $5n^2 \log_2 n + 2n^2 = O(n^2 \log_2 n)$
   $c \geq 7, n_0 = 2$

7. Watch the video lecture on Big O, Big Ω and Big Θ notation from
   http://www.youtube.com/watch?v=6Ol2JbwoJp0. Write the summary of the lecture in your words. [10 Points]

   Solution. Give full marks if they write something about Big O, Big Ω and Big Θ notation

8. Use Master Theorem, to calculate the time complexity of the following [15 points]

$$T(n) = \sqrt{2}\, T(n/2) + \log n \quad (1)$$
$$T(n) = 64\, T(n/8) - n^2 \log n \quad (2)$$
$$T(n) = 16T(n/4) + n! \quad (3)$$

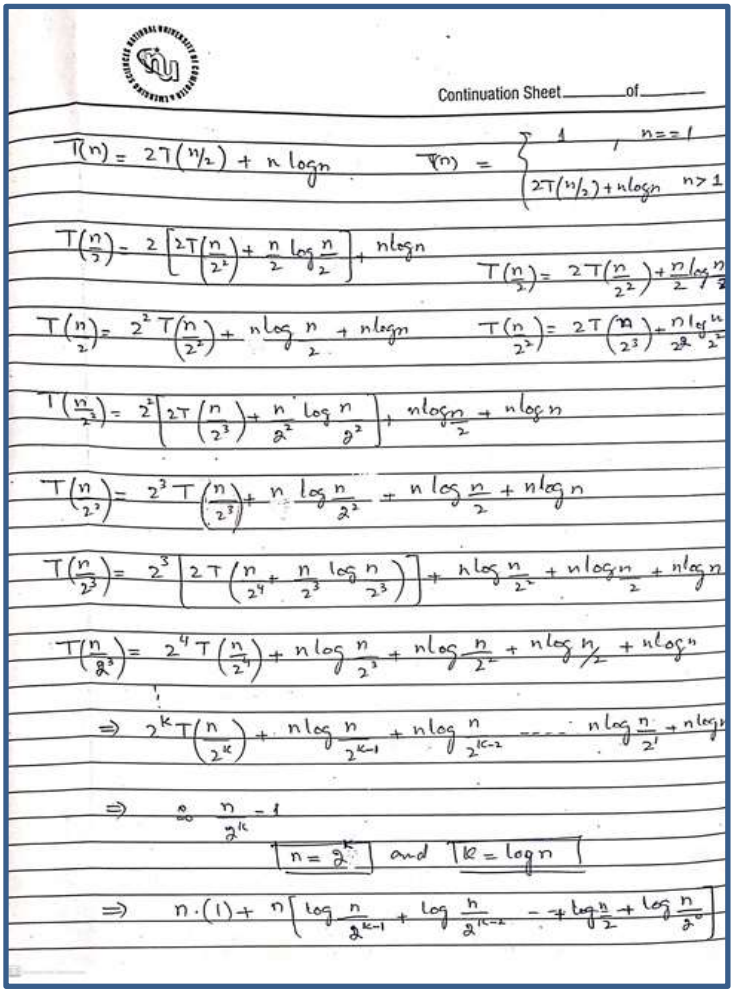
Sol:

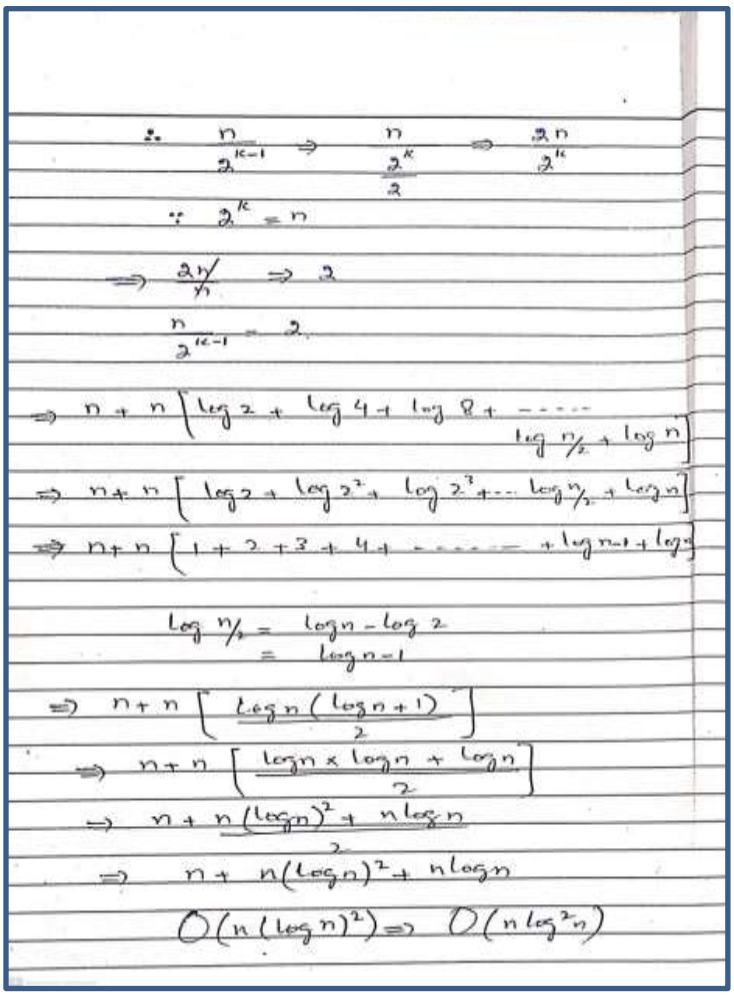
Not possible for all three

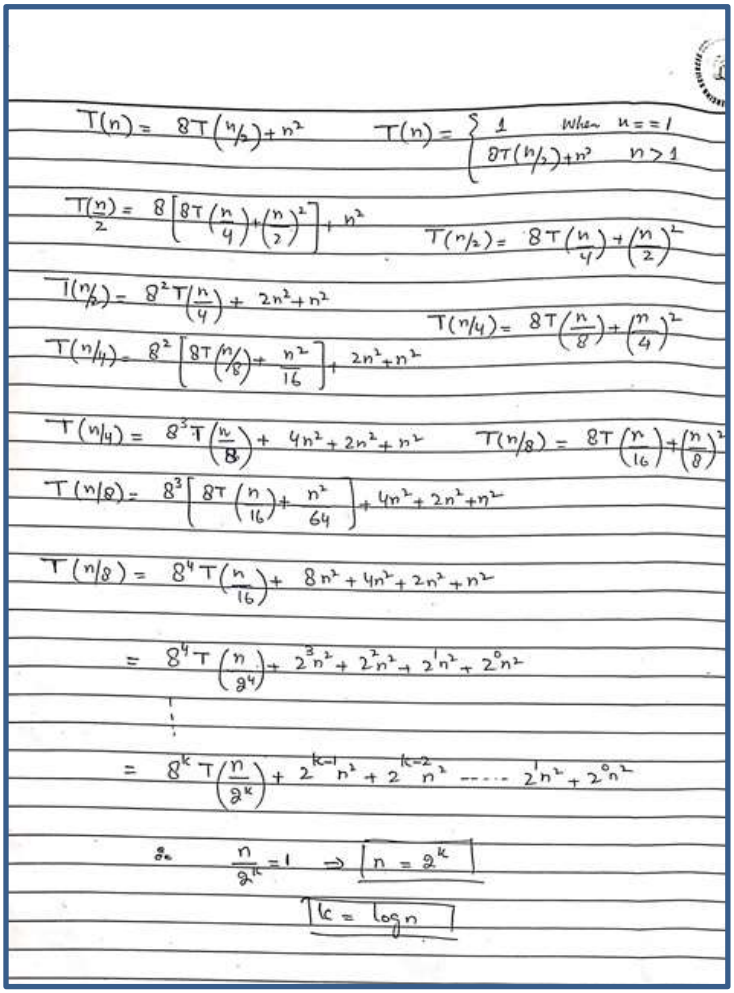
9. Use Iteration Method, to calculate the time complexity of the following [10 points]

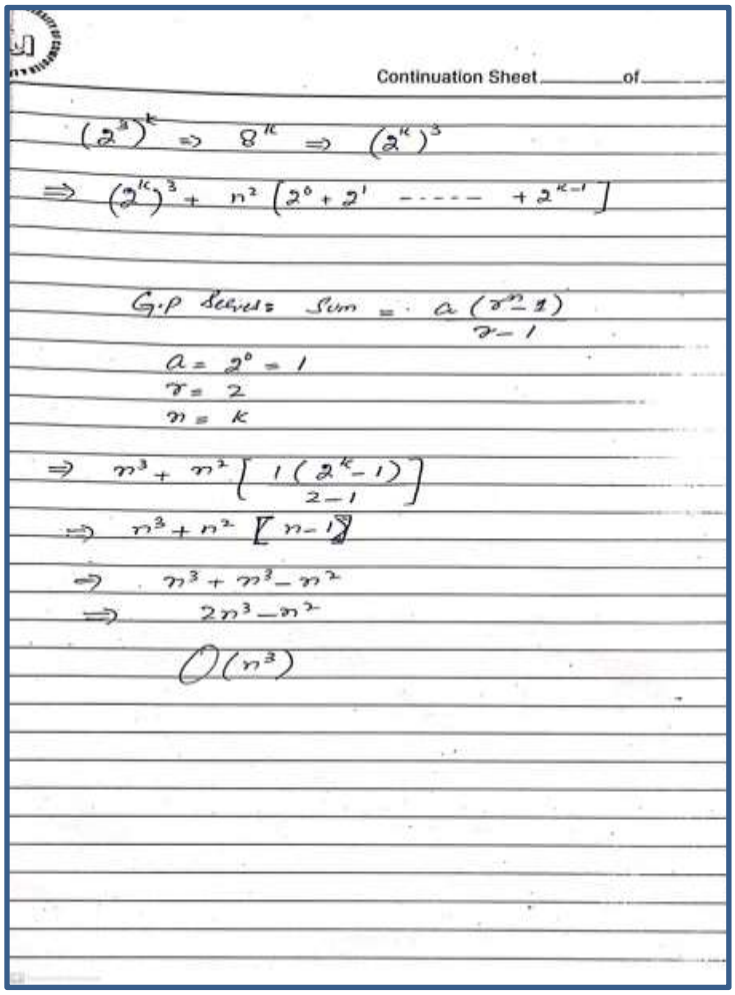
$$T(n) = 2T(n/2) + n\log n \quad (4)$$

$$T(n) = 8T(n/2) + n^2 \quad (5)$$




∴ $\frac{n}{2^{k-1}} \Rightarrow \frac{n}{\frac{2^k}{2}} \Rightarrow \frac{2n}{2^k}$
$\because 2^k = n$
$\Rightarrow \frac{2n}{n} \Rightarrow 2$
$\frac{n}{2^{k-1}} = 2$
$\Rightarrow n + n\left[\log 2 + \log 4 + \log 8 + \cdots \log \frac{n}{2} + \log n\right]$
$\Rightarrow n + n\left[\log 2 + \log 2^2 + \log 2^3 \cdots \log \frac{n}{2} + \log n\right]$
$\Rightarrow n + n\left[1 + 2 + 3 + 4 + \cdots + \log n - 1 + \log n\right]$
$\log \frac{n}{2} = \log n - \log 2$
$= \log n - 1$
$\Rightarrow n + n\left[\frac{\log n(\log n + 1)}{2}\right]$
$\Rightarrow n + n\left[\frac{\log n \times \log n + \log n}{2}\right]$
$\Rightarrow n + \frac{n(\log n)^2 + n \log n}{2}$
$\Rightarrow n + n(\log n)^2 + n \log n$
$O(n(\log n)^2) \Rightarrow O(n \log^2 n)$


T(n) = 8T(n/2) + n²
T(n) = { 1 when n = 1
8T(n/2) + n² n > 1
T(n/2) = 8T(n/4) + (n/2)²
T(n/4) = 8T(n/8) + (n/4)²
T(n/8) = 8T(n/16) + (n/8)²
= 8^k T(n/2^k) + 2^{k-1} n² + 2^{k-2} n² ---- 2¹n² + 2⁰n²
n/2^k = 1 ⇒ n = 2^k
k = log n



Continuation Sheet ____ of ____

$(2^3)^k \Rightarrow 8^k \Rightarrow (2^k)^3$

$\Rightarrow (2^k)^3 + n^2 [2^0 + 2^1 ----- + 2^{k-1}]$

G.P Series: Sum $= \frac{a(r^n - 1)}{r - 1}$

$a = 2^0 = 1$

$r = 2$

$n = k$

$\Rightarrow n^3 + n^2 \left[\frac{1(2^k - 1)}{2 - 1}\right]$

$\Rightarrow n^3 + n^2 [n - 1]$

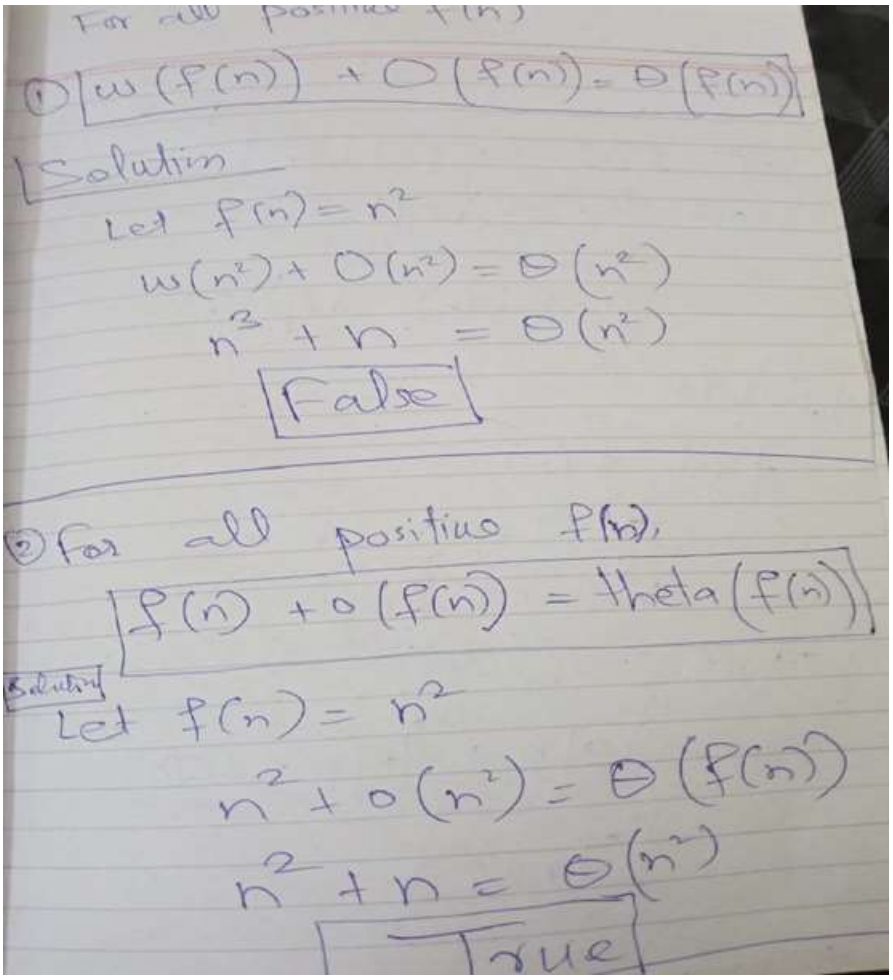
$\Rightarrow n^3 + n^3 - n^2$

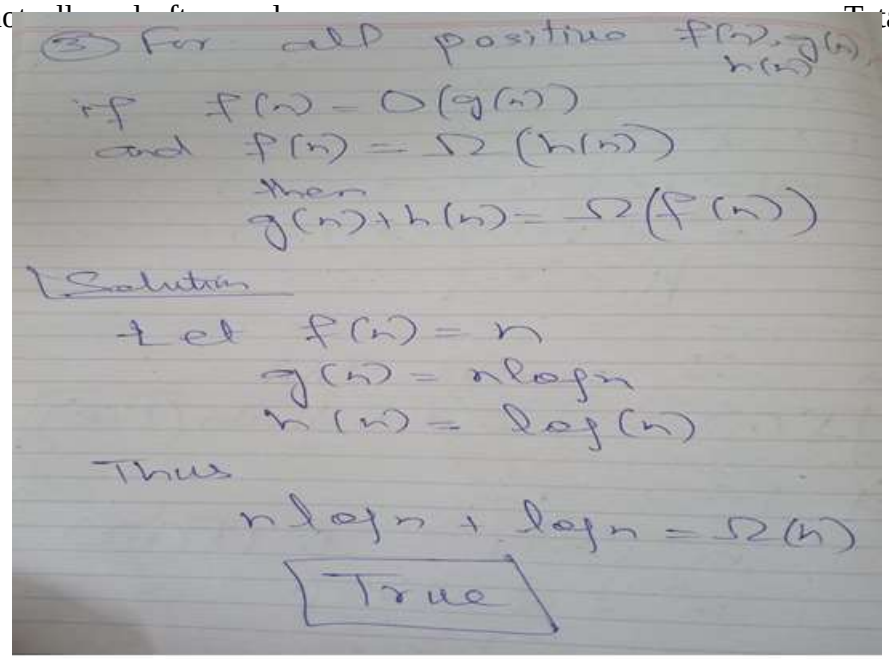
$\Rightarrow 2n^3 - n^2$

$O(n^3)$

10. For each of the following questions, indicate whether it is T (True) or F (False) and justify using some examples e.g. assuming a function? [15 Points]

- For all positive f(n), g(n) and h(n), if $f(n) = O(g(n))$ and $f(n) = \Omega(h(n))$, then $g(n) + h(n) = \Omega(f(n))$.
- Let f(n) and g(n) be asymptotically nonnegative functions, then max (f(n), g(n)) $= \Theta$ (f(n) + g(n)).
- if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$, then we have $(f(n))^2 = (g(n))^2$




③ For all positive f(n), g(n), h(n)
if f(n) = O(g(n))
and f(n) = Ω(h(n))
then
g(n) + h(n) = Ω(f(n))
Solution
Let f(n) = n
g(n) = nlogn
h(n) = log(n)
Thus
nlogn + logn = Ω(n)
True

Due Date: 20 September 2022
20% penalty for 1 day late
40% penalty for 2 days late
Submission not allowed afterwards

CS2009: Design and Analysis of Algorithms (Fall 2023)
Assignment 1
Total Marks: 100