1. What is the difference between $O(g(n))$ and $\Theta(g(n))$?

While $O(g(n))$ represents class of functions $f(n)$ that grow no faster than $g(n)$, whereas $\Theta(g(n))$: class of functions $f(n)$ that grow at same rate as $g(n)$

2. True False Questions:

1	F	An algorithm with the time complexity O(n) is faster than algorithm with the time complexity O(nlogn), where both algorithms run on the same machine and n=100.
2	F	QuickSort is faster than binary search for the same input size N.
3	T	Worst case time complexity of sequential/linear search is O(n)
4	T	Big-Oh Notation shows the worst case time complexity.
5	T	If the given problem can be solved by using permutation, then time complexity of the given problem is in the range of factorial (O(n!)).
6	F	An algorithm with time complexity $O(n^2)$ is faster than the algorithm O(nlogn)
7	F	Time complexity of TSP problem is $O(2^n)$).
8	T	Time complexity of KnapSack problem is O(n!).
9	T	Insertion Sort is an example of Decrease-and Conquer approach.
10	T	Searching with presorting can be done in $\Theta(n\log n) + O(\log n) = \Theta(n\log n)$

3. Solve the following recurrences using the master method.

$T(n) = 2T(n/2) + n^3 \Rightarrow T(n) \in ?$
$T(n) = 16T(n/4) + n^2 \Rightarrow T(n) \in ?$
$T(n) = 7T(n/2) + n^3 \Rightarrow T(n) \in ?$

4. Suppose we are comparing implementations of algorithms *A* and *B* on the same machine. For inputs of size *n*, *A* runs in $5n^2$milliseconds, while *B* runs in 2^nmilliseconds. For which values of *n* does *A* beat *B*? Show your calculations clearly.

SOLUTION:

N	$5n^2$	2^n	$5n^2$ vs 2^n
1	5	2	$5n^2 > 2^n$
2	20	4	$5n^2 > 2^n$
3	45	8	$5n^2 > 2^n$
4	80	16	$5n^2 > 2^n$
5	125	32	$5n^2 > 2^n$

6	180	64	$5n^2 > 2^n$
7	245	128	$5n^2 > 2^n$
8	320	256	$5n^2 > 2^n$
9	**405**	**512**	$\mathbf{5n^2 < 2^n}$
10	**500**	**1024**	$\mathbf{5n^2 < 2^n}$
11	**…**	**…**	$\mathbf{5n^2 < 2^n}$
…	**…**	**…**	$\mathbf{5n^2 < 2^n}$

For N>= 9, $5n^2 < 2^n$=> *the algorithm A with complexity* $5n^2$ beats (here, we mean A is faster than B) *the algorithm B with complexity* 2^n