1. Consider the recurrence

$$T(n) = \begin{cases} 9T(n/3) + \mathcal{O}(n^2) & \text{for } n > 1 \\ 1 & \text{for } n = 1. \end{cases} \tag{1}$$

   Solve the recurrence to find a tight upper bound by using substitution method.

2. An algorithm solves a problem of size n by dividing it into 5 sub-problems, each of which has the size of $n/2$, and it recursively solves each of these 5 sub-problems. Then it combines solutions of sub-problems to form the solution of the initial problem in linear time ($\mathcal{O}(n)$).
   (a) Write the recurrence of this algorithm.
   (b) Solve the recurrence to find a tight bound for the recurrence by using a recursion tree.

3. For the below functions, show which of the asymptotic bounds hold for $f(n)$.
   1. $\mathcal{O}(g(n))$
   2. $\Omega(g(n))$
   3. Both (i.e. $\Theta(g(n))$)

   In order to get full-credit, you need to find constants c and n_0 such that $f(n) \le c \cdot g(n)$ for all $n \ge n_0$, if you think that $f(n)$ is $\mathcal{O}(g(n))$. Similarly, find c and n_0 such that $f(n) \ge c \cdot g(n)$ for all $n \ge n_0$, if $f(n)$ is $\Omega(g(n))$. (NOTE: Every log is base 2.)
   (a) $f(n) = 3n^2 \qquad g(n) = n^2$
   (b) $f(n) = 2n^4 - 3n^2 + 7 \qquad g(n) = n^5$
   (c) $f(n) = \frac{\log n}{n} \qquad g(n) = \frac{1}{n}$
   (d) $f(n) = \log n \qquad g(n) = \log n + \frac{1}{n}$
   (e) $f(n) = 2^{k \log n} \qquad g(n) = n^k$
   (f) $f(n) = 2^n \qquad g(n) = 2^{2^n}$
   (g) $f(n) = 2^{\sqrt{\log n}} \qquad g(n) = (\log n)^{100}$
   (h) $f(n) = \begin{cases} 4^n & n < 2^{1000} \\ 2^{1000} n^2 & n \ge 2^{1000} \end{cases} \qquad g(n) = \frac{n^2}{2^{1000}}$

4. State the runtime recurrence if it is not given. Then find the asymptotic running time (big-$\mathcal{O}$) of the following algorithms using the Master theorem. If it is not solvable using Master theorem, explain your reasoning.
   (a) $T(n) = 3T(n/4) + \mathcal{O}(n)$
   (b) An algorithm solves a problem of size n by dividing it into 2^n sub-problems of size $n/2$, and recursively solves corresponding sub-problems. It takes linear time, $\mathcal{O}(n)$ to combine solutions of the sub-problems.