1. (10 points) Consider the function $f(n) = 2^{2n}$.
   
   (a) Is it $O(2^n)$? Prove your answer.
   (b) Is it $\Omega(2^n)$? Prove your answer.

2. (15 points) Prove that $f(n) = \Theta(g(n))$ if and only if $g(n) = \Theta(f(n))$.

3. (20 points) A function $f(n)$ is said to be $o(g(n))$ (pronounced “little-oh”), if for any positive constant $C$, there exists a positive constant $k$ such that:

   $\forall n > k. \quad f(n) < C \cdot g(n)$

   (a) Consider the function $f(n) = n^2$ where the domain of $f$ is positive integers. Is it $o(n^2)$? Prove your answer.
   (b) Consider the same function $f(n) = n^2$ where the domain of $f$ is positive integers. Is it $o(n^3)$? Prove your answer.
   (c) Suppose a function $h(n)$ is $o(g(n))$. Is $g(n)$ always $\Omega(h(n))$? Prove your answer.
   (d) Suppose a function $h(n)$ is $\Theta(g(n))$. Is it possible that $h(n)$ is $o(g(n))$? Prove your answer.

4. (10 points) A vending machine in Europe accepts either 1 Euro bills, 1 Euro coins, or 2 Euro bills. Assume that the order in which money is inserted into the machine matters (i.e., inserting 1 Euro bill followed by 1 Euro coin is different from inserting 1 Euro coin first and then a 1 Euro bill). Let $a_n$ denote the number of ways of inserting $n$ Euros into the vending machine.
   
   (a) Determine the values of $a_1$ and $a_2$. 

(b) Write a recurrence relation describing \( a_n \).

(c) Determine the closed form solution for \( a_n \).

5. (10 points) Find the closed form solution to \( a_n = 7a_{n-2} + 6a_{n-3} \) for \( a_0 = 9 \) with initial values \( a_1 = 10 \) and \( a_2 = 32 \).

6. (10 points) Find a particular solution for the recurrence \( a_n = 2a_{n-1} + 3a_{n-2} + 3^n \).

7. (10 points) Solve the recurrence \( a_n = a_{n-1} + n \) with initial condition \( a_0 = 1 \).

8. (15 points) Consider a “ternary search” algorithm, which is a variation on binary search. In particular, the ternary search algorithm works as follows:

- It takes as input a sorted array \( a \) of size \( n \) and an integer \( i \) to search for
- If \( n = 0 \), it returns false; and if \( n = 1 \), it returns true if the only element of \( a \) is \( i \)
- If \( i \leq a\lfloor n/3 \rfloor \), then, it searches the subarray \( a[0...n/3] \)
- If \( i > a\lfloor n/3 \rfloor \) and \( i \leq a\lfloor 2n/3 \rfloor \), then it searches the subarray given by \( a\lfloor (n/3 + 1)...2n/3 \rfloor \)
- Otherwise, searches the subarray \( a\lfloor (2n/3 + 1)...n \rfloor \)

Let \( T(n) \) describe the number of operations performed by ternary search for an input array of size \( n \).

(a) Write a recurrence relation describing \( T(n) \) and give the initial value for \( n = 1 \).

(b) Find a closed form solution for this recurrence assuming that \( n \) is a power of 3.

(c) Use the Master Theorem to obtain a Big-Theta estimate for \( T(n) \)