Part 1

1. True

f(n) = O(g(n)) -> there are positive constants c and n0,such that 0 <= f(n) <= cg(n) for all n >= n0.

Proof:

n^13 = O(n^10)
=> n^13 <= c(n^10)
=> n^3 <= c

c must be a constant for it to satisfy Big-O
therefore for n^13 to be O(n^10), c must e greater than n^3
since c depends on n, we can confirm that the statement is in fact true.

2. Yes, T(n) has a Big-theta estimation

given, T(n) = a(k)*n^k + a(k-1)*n^(k-1) + ... + a(1)*n + a(0)
in order to prove this claim, we must find the upper and lower bounds.

Upper Bound:

Let's say that:

T(n) <= max(a(k), a(k-1), ... , a(0)) * n^k + max(a(k), a(k-1), ... , a(0)) * n^k + max(a(k), a(k-1), ... , a(0)) * n^k + max(a(k), a(k-1), ... , a(0)) * n^k
for some n >= 1

Then:

T(n) <= n * max(a(k), a(k-1), ... , a(0)) * (n^k)

Thus, we can assign a constant 'c' as follows:

c = max(a(k), a(k-1), ... , a(0))

Therefore:

T(n) <= c * n^(k+1)

Lower Bound:

Let's say that:

T(n) >= min(a(k), a(k-1), ... , a(0)) * (n/2)^k + min(a(k), a(k-1), ... , a(0)) * (n/2)^k + min(a(k), a(k-1), ... , a(0)) * (n/2)^k + min(a(k), a(k-1), ... , a(0)) * (n/2)^k
for some n >= 1

Then:

T(n) >= n * min(a(k),a(k-1), ... , a(0)) * (n^k)/(2^k)

Thus, we can assign a constant 'c' as follows:

c = min(a(k), a(k-1), ... , a(0))

Therefore:

T(n) >= c * n^(k+1)

Since we have proven both the upper and lower bound, we can conclude that: T(n) = theta(n^(k+1))

3. Given, h(x) is O(g(x))

Thus, there exists a real number 'n' and a positive real number 'p', such that:

for x > n , |h(x)| <= p * |g(x)|

Given, g(x) is O(f(x))

Thus, there exists a real number 'm' and a positive real number 'c', such that:

for x > m , |g(x)| <= c * |f(x)|

Therefore, if x > 2 = max(n, m)

|h(x)| <= p * |g(x)| and |g(x)| <= c * |f(x)|

Then:

for all x > 2, |h(x)| <= p * |g(x)| <= c * |f(x)|

Based on the data above, we can conclude that h(x) is O(f(x))