Recurrences, Master Theorem, tree and table method, induction.

1. Given the recurrences
    a. T(N) = 3*T(N/5) + N + lgN
    b. T(N) = 4*T(N/2) + $\sqrt{N}$
    c. T(N) = 6*T(N/5) + $N^3$
    d. T(N) = 6*T(N/5) + 7

    Find their Θ time complexity with the tree method. <u>You must show the tree and fill out the table like we did in class.</u>
    Find their Θ time complexity with the Master Theorem method.
2. Use the substitution method (induction) to show that T(N) = 2T(N/2) + $N^3$ is O($N^3$). Let T(0)=4.
3. CLRS 3rd edition (textbook)
    a. Reminder: The book calls 'substitution method' what we called 'induction method'.
    b. Page 87: 4.3-1 – Consider every one of the three methods. Can you apply it? If yes, solve with that method, if no, explain why.
    c. Page 87, 4.3-7
    d. page 92, 4.4-1, 4.4-2, 4.4-3 (NOT with the tree on the given recurrence. Instead, use a similar but easier recursion, and guess it with the Master theorem or the tree and prove it with induction).
    e. page 96, 4.5-1

**41. (6 points)** A recursive algorithm for processing arrays works as follows: it first does some processing which takes $N^2$ and allows it to split the array in 3 equal parts. Next the algorithm applies itself again to each one of those smaller arrays.

If the array has 0, 1, or 2 elements the algorithm executes 5 instructions and finishes. Give the recurrence formula (including the base case) for this algorithm.

**P5 . (Exam 1, Fall 15, 002)**

a) (5 points) Is anything wrong with the following recurrence definition?

g(0) = N

g(N) = g(N-1) + c

**P6.** (Exam 1, Fall 15, 002)

```
int foo(int * array, int N)
{
  if (N == 0) return 0;
  int result = 0;
  int b, c;
  for (b = 0; b < N/4; b++)
    for (c = N; c > 1 ; c = c/2)
      result = result + array[b] * array[c];
  return result + foo(array, N-1);
}
```

Give the recurrence formula (including the base case).