Recurrences, Master Theorem, tree and table method, write the recurrence for recursive functions.

Conventions: When giving answers (online):

- Do NOT put any spaces in your answers
- For level TC (time complexity) give the answer in the form: number_of_nodes*1_node_TC without any spaces. E.g. the tree for the recurrence **T(n) = 4T(n/2) + cn** has at level TC at level t: **4^t*c(n/2^t)** where 4^t is the number of nodes per level and c(n/2^t) is the TC of one node at level t.

P1. 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.

P2. Solve the recurrence T(n) = 2T(n-3)+c, where T(n) = c for all n≤3.

P3. Can you solve the recurrence: $T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + cn$ with base cases $T(N) = c\ for\ N \leq 13$? Here the symbols ⌊ ⌋, ⌈ ⌉ indicate rounded and down and rounded up.

P4. (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) (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).

P7. Short answer.

a) (5pts) Can you apply the master theorem for the recurrence **T(n) = 4T(n-2) + cn**? Justify. (E2,Fall 18)

b) (6 pts) Consider the tree for the recurrence **T(n) = 4T(n/2) + cn**. Fill in the answers regarding the tree for this recurrence:

   Any internal node has _____ children.

   The **problem size** for a node on level 3 is ________ (where the root is at level 0).

   The **TC of a single** node on level 3 is ________ (where the root is at level 0).

   The **last level** is (if the answer involves a log, indicate the base for it) : k = ________

   How many nodes will the tree have at level t ? __________

   What is the **Level TC** at some level t? ____________

c) (5 pts) Give a recurrence formula that will result in a tree that has the last level: k = N/4

d) *This problem if not covered in Fall 2020.* (4 pts) Mark (with X) the correct statement about the Tree (and table) method for solving recurrences as done in class:

   _____ it computes an estimate of the time complexity (but it does not completely prove it)

   _____ it computes and mathematically proves the correct answer.

e) (5 pts) Consider the recurrence: T(N) = T(N-7)+c. Assume the first applicable value for N is 0 (i.e. assume it is never applied to negative values).

   How many base case(s) does this problem have? (2pts) : _________________

   List the values of N for the base case (3pts) : _______________________

f) T/F: 3^{log_2(N)} = N

P8. (10pts) Can you use the **Master theorem** to solve the recurrence: **T(n) = 4T(n/2) + n** ? If yes, solve it with this method (make sure you indicate the case ~~give the value for ε where needed and use limit theorem for O/Θ/Ω~~). If no, show why you cannot use it.

T(n) = 4T(n/2) + n

P9. (8 pts) Write a recursive function that has the recurrence formula (for time complexity): T(N) = 2*T(N/3) + cN and base cases: T(N) = c (for all N≤2).

P10. (30 pts) Use the tree method to compute the **Θ** time complexity for T(N) = 4T(N/4) + cN with T(1) = c .

Fill in the table below and finish the computations outside of it:

Level	Argument/ Problem size	Cost of one node	Nodes per level	Cost of whole level
0				
1				
2				
l				
k= *Leaf level.* *Write k as a function of N.*				

Total tree cost calculation: ……………………………………………………………….

T(N) = Θ(…………………)

Draw the tree.-Show **levels 0,1 and 2.** (Show just a few nodes at level 2) Show the problem size T(…) as a label next to the node and inside the node show the local cost (cost of one node) as done in class.

EXTRA: topic induction method (not covered and not required for test or quiz)

PExtra1. Use the substitution method (induction) to show that $T(N) = 2T(N/2) + N^3$ is $O(N^3)$. Let $T(0)=4$.

PExtra2. 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 (This only requires Master Theorem)**