The test itself will not be as long.

1.) Induction
2.) Asymptotic Analysis
3.) Run-time Code analysis.
4.) Divide and Conquer
    a.) Substitution
    b.) Recurrence tree
    c.) Masters

Proof by induction

$$\sum_{i=1}^{n+1} i \cdot 2^i = n2^{n+2} + 2 \qquad \text{For all integer n >= 0}$$

Solution:

Base cases: n=0, L.S.= 2 =R.S.

Assume f(k) is true, i.e $\sum_{i=1}^{k-1} i2^i = k2^{k+2} + 2$

When n = k+1 $\sum_{i=1}^{k+2} i2^i$

$$
\begin{aligned}
&= (k+2)2^{k+2} + \sum_{i=1}^{k-1} i2^i \\
&= (k+2)2^{k+2} + k2^{k+2} + 2 \\
&= k2^{k+2} + 2^{k+3} + k2^{k+2} + 2 \\
&= k2^{k+3} + 2^{k+3} + 2 \\
&= (k+1)2^{k+3} + 2
\end{aligned}
$$

Asymptotic Analysis

2.) 5pts

By definition or limit lemma

Find c and M to prove that $2n^5 + 3n^3 + 6 = O(n^5)$.

Solution:

Note $2n^5 + 3n^3 + 6n \leq 2n^5 + 3n^5 + 6n^5$ for $n \geq 1$

$= 11n^5$

So picking $c = 11$, $M = 1$ shows

$2n^5 + 3n^3 + 6 = O(n^5)$

3.) Divide & Conquer solving recurrences and code analysis.

- **Code analysis is also an option for this question like on the quizes..**

T(n)= T(n/3)+n

a.) By recursion tree method (4 pts)

Provide the tightest bound for **T(n)= T(n/3)+n**

$T(n) = T(n/3) + n$

n

$T(n/3)$ $\quad n/3$

$T(n/9)$ $\quad n/9$

I see the pattern that for level i there is $\frac{n}{3^i}$ amount of operations

the depth is:

$\frac{n}{3^i} = 1$ ← BASE CASE

depth $= i = \log_3 n$

$\sum_{i=0}^{\log_3 n} n/3^i = n \sum_{i=0}^{\log_3 n} \frac{1}{3^i} \Rightarrow$ geo $\frac{1}{1-a} = \frac{1}{2/3} = 3/2$

$= n(3/2) \Rightarrow O(n)$

b.) By substitution (3 pts)

Prove the above with substitution from the solution you found in a c.) By Master theorem (2pts)

$T(n) = T(n/3) + n$

$T(n) \le cn$

$T(n) \le c(\frac{n}{3}) + n \le cn$

$c(\frac{n}{3}) + n \le cn$

$\frac{c}{3} + 1 \le c$

$1 \le 2/3\,c$

$3/2 \le c$ $\quad c = 2$

$T(n) = T(n/3) + n$

$f(n) = \Theta(n^1)$

$a = 1 \quad b = 3$

$\log_b a = \log_3 1 \ ? \ 1$

$<$

CASE 3

$\Theta(n)$

4.) **Divide and Conquer analysis**

a.)

. Suppose you are choosing between the following three algorithms:

- Algorithm A solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.
- Algorithm B solves problems of size n by recursively solving two subproblems of size $n-1$ and then combining the solutions in constant time.
- Algorithm C solves problems of size n by dividing them into nine subproblems of size $n/3$, recursively solving each subproblem, and then combining the solutions in $O(n^2)$ time.

What are the running times of each of these algorithms (in big-O notation), and which would you choose?

ANSWER: Best algorithm (fastest) asymp in worst case: Algo. C, Algo A, ~~log~~ Algo B
~~Mesmgh~~

WORK:

Algo A $5T(n/2) + O(n)$ $\Rightarrow$ $O(n^{2.32})$

Algo B $2T(n-1) + O(1)$ $O(2^n)$

Algo C $9T(n/3) + O(n^2)$ $O(n^2 \log n)$

How did I come up with the answer:

Algo A: use MASTERS THM CASE 1

$O(n^{\log_2 5}) \approx O(n^{2.32})$

Algo B: MASTERS DOES NOT WORK, USE TREE METHOD

Tree	WORK AT LEVEL
$\circ$	1
$T(n-1)$ $T(n-1)$	2
$T(n-2)$ $T(n-2)$ $T(n-2)$ $T(n-2)$	2^2

work at level i

i : 2^i

depth

$n - i = 1$

$i = n-1$

$$\sum^{\text{depth}} \text{work at level} = \sum_{i=0}^{n-1} 2^i = 2^{(n-1)+1} - 1 = 2^n - 1$$

Algo C: MASTERS THEOREM CASE 2

$O(n^2 \log n)$

How do I know that n^log_2(5) >n^2*logn

proof: algo C is smaller asympt. then Algo A

$$\lim_{n \to \infty} \frac{n^2 \log n}{n^{2.32}} = \lim_{n \to \infty} \frac{\log n}{n^{0.32}}$$

B.C. its $\frac{\infty}{\infty}$ we must use l hopes rule

$$\lim_{n \to \infty} \frac{\log n}{n^{0.32}} = \frac{1}{n^{1}} \cdot \frac{1}{n^{-.78}} = \frac{1}{n^{.32}} = \infty$$

b.)
Examining algorithms from HW2 problem 8 and 9