Dennis Kuzminer
CSCI-UA 310-001 PS1

1.

a. $\lim\limits_{n\to\infty} \frac{n(log_2 n)^2}{n^2 log_2 n} \to \lim\limits_{n\to\infty} \frac{log_2 n}{n} \to$ D.S. = $\frac{\infty}{\infty} \to$ L'Hopital $\to \frac{\frac{1}{nln(2)}}{1} \to \frac{1}{nln(2)} \to$ D.S. = $\frac{1}{\infty} = 0$ $\to$ This implies that g grows faster than f and **f=o(g)**

b. $\lim\limits_{n\to\infty} \frac{n^2}{n(log_2 n)} \to \lim\limits_{n\to\infty} \frac{n}{(log_2 n)} \to$ D.S. = $\frac{\infty}{\infty} \to$ L'Hopital $\to \frac{1}{\frac{1}{nln(2)}} \to nln(2) \to$ D.S. = ∞ $\to$ This implies that f grows faster than g and **g=o(f)**

c. $\lim\limits_{n\to\infty} \frac{n(log_2 n)^4}{n^{1.2}} \to \lim\limits_{n\to\infty} \frac{(log_2 n)^4}{n^{.2}} \to$ D.S. = $\frac{\infty}{\infty} \to$ L'Hopital $\to \lim\limits_{n\to\infty} \frac{4(log_2 n)^3}{.2n^{.2} ln(2)} \to$ L'Hopital $\to$ $\lim\limits_{n\to\infty} \frac{12(log_2 n)^2}{.2n^{.2} ln(2)ln(2).2} \to$ L'Hopital $\to \lim\limits_{n\to\infty} \frac{24(log_2 n)}{.2n^{.2} ln(2)ln(2).2ln(2).2} \to$ L'Hopital $\to$ $\lim\limits_{n\to\infty} \frac{24}{.2n^{.2} ln(2)ln(2).2ln(2).2ln(2).2} \to$ D.S. = 0 $\to$ This implies that g grows faster than f and **f=o(g)**

d. $\lim\limits_{n\to\infty} \frac{200n^2+n^{1.5}}{(1/500)n^2} \to \lim\limits_{n\to\infty} \frac{200n+n^{.5}}{(1/500)n} \to$ D.S. = $\frac{\infty}{\infty} \to$ L'Hopital $\to \frac{200 + .5/n^{.5}}{(1/500)} \to$ D.S. = $\frac{200+0}{(1/500)} = 200 * 500 = 100000 \to$ This implies that f grows at the same rate as g and **f = θ(g)**

e. $\lim\limits_{n\to\infty} \frac{log_7 n}{log_5 n} \to$ D.S. = $\frac{\infty}{\infty} \to$ L'Hopital $\to \frac{\frac{1}{nln(7)}}{\frac{1}{nln(5)}} \to \frac{nln(5)}{nln(7)} \to$ D.S. = $\frac{\infty}{\infty} \to$ L'Hopital $\to$ $\frac{ln(5)}{ln(7)} \to$ This implies that f grows at the same rate as g and **f = θ(g)**

f. $\lim\limits_{n\to\infty} \frac{n(log_2 n)^{-1}}{n^{.5} log_2 n} \to \lim\limits_{n\to\infty} \frac{n^{.5}}{(log_2 n)^2} \to$ D.S. = $\frac{\infty}{\infty} \to$ L'Hopital $\to \frac{.5n^{.5} ln(2)}{2log_2 n} \to$ D.S. = $\frac{\infty}{\infty} \to$ L'Hopital $\to \frac{.5ln(2)}{2} \frac{n^{.5}}{log_2 n} \to \frac{.5ln(2)}{2} \frac{.5/n^{.5}}{1/nln(2)} \to \frac{.5^2 ln(2)^2}{2} \frac{n}{n^{.5}} \to \frac{.5^2 ln(2)^2}{2} n^{.5} \to$D.S. $\to \infty \to$ This implies that f grows faster than g and **g=o(f)**

g. $\lim\limits_{n\to\infty} \frac{5^n}{7^n} \to \lim\limits_{n\to\infty} (\frac{5}{7})^n \to \lim\limits_{n\to\infty} e^{nln(\frac{5}{7})} \to \lim\limits_{n\to\infty} e^{n(-0.336472237)} \to \lim\limits_{n\to\infty} 1/e^{n(0.336472237)} \to$ D.S. = 0 $\to$ This implies that g grows faster than f and **f=o(g)**

h. $\lim\limits_{n\to\infty} \frac{7^n}{5^{(n^2)}} \to \lim\limits_{n\to\infty} (\frac{7}{5^n})^n \to$ D.S. = $(\frac{7}{\infty})^\infty \to (0)^\infty = 0 \to$This implies that g grows faster than f and **f=o(g)**

i. $\lim\limits_{n\to\infty} \frac{n!}{(n+1)!} \to \lim\limits_{n\to\infty} \frac{1}{(n+1)} \to$ D.S. = $\frac{1}{\infty} = 0 \to$ This implies that g grows faster than f and **f=o(g)**

j. $\lim\limits_{n\to\infty} \frac{n}{2n+(-1)^n n^{.5}} \to$ Divide all terms by n $\to \lim\limits_{n\to\infty} \frac{1}{2+(-1)^n n^{-.5}} \to (-1)^n n^{-.5} = 0$ as n$\to \infty$. Therefore, $\lim\limits_{n\to\infty}$ = ½ $\to$ This implies that f grows at the same rate as g and **f = θ(g)**

Dennis Kuzminer
CSCI-UA 310-001 PS1

2.

a. $\int_1^n ln(i)di \rightarrow [iln(i) - i]_1^n \rightarrow nln(n) - n - (1ln(1) - 1) = nln(n) - n + 1$ which is essentially $= nln(n) + O(n)$

b. $\int_1^n iln(i)di \rightarrow IBP \rightarrow u = ln(i),\ v' = i \rightarrow \left[.5i^2 ln(i) - \int .5idi \right]_1^n \rightarrow$
$.5n^2 ln(n) - .25n^2 - (5(1)^2 ln(1) - .25(1)^2) \rightarrow .5n^2 ln(n) - .25n^2 + .25\ = .5n^2 ln(n) + O(n^2)$

Dennis Kuzminer
CSCI-UA 310-001 PS1

3. $L = \lim_{i\to\infty} \frac{(i+1)^2}{2^{(1+i)}} * \frac{2^i}{i^2} \to \frac{i^2+2i+1}{2i^2} \to L'H \to \frac{2i+2}{4i} \to L'H \to \frac{2}{4} \to \frac{1}{2}$

   L is less than 1; therefore, the infinite series **converges** absolutely.

Dennis Kuzminer
CSCI-UA 310-001 PS1

4.

a. $\int_{1}^{\infty} \frac{1}{i^{1.1}} di \rightarrow -10(1/i^{.1}) \rightarrow -10(1/\infty^{.1}) - -10(1/1^{.1}) = 0 -- 10 = 10$

Because the integral converges, we can also say that the series **converges**.

b. $\int_{2}^{\infty} \frac{1}{i ln(i)} di \rightarrow ln|ln(i)| \rightarrow ln|ln(\infty)| - ln|ln(2)| \rightarrow \infty -- .3665 =$ Divergent.

Because the integral diverges, we can also say that the series **diverges**.

Dennis Kuzminer
CSCI-UA 310-001 PS1

5.

a. The first for-loop creates a runtime of n. The second for-loop creates a runtime of n again, meaning that the first for-loop's run time will not matter as $n \rightarrow \infty$. The while-loop inside of the for-loop will consistently reduce the number of operations that need to be done in comparison O(n). For instance, if n = 10, then the first iteration of the for-loop will perform 10 (* 2) operations within the while-loop. In the second iteration, 5 (* 2). Then 3 (* 2), 2 (* 2), 2 (* 2), 1 (* 2), and so on. We can see that the number of iterations/operations of the while-loop is decreasing by some proportion of n as n grows. Therefore, we can simplify the runtime of this algorithm to n+nlogn. We can simplify, we get **O(nlogn)**.

b. If we run the program for an array of integers size 12, we will be adding one to indices A[i] every for-loop iteration. However, we do not add it to every index. We add one to every ith index. For instance, if i = 4, then we would only add one to A[4], A[8], and A[12]. At the end of the 12th iteration of the algorithm, we can see that A = [1,2,2,3,2,4,2,4,3,4,2,6]. Each value of A[i] actually **represents the number of factors of a given number i**.

Dennis Kuzminer
CSCI-UA 310-001 PS1

6. n=leaves and m=internal nodes
   Q0 → For a 2-3 tree of height zero, n = 1 and m = 0; therefore, the assertion holds for h = 0, as 0 ≤ 0.
   Inductive step: Assume that m≤n-1holds for all h-1 ≥ 0 → Show it holds for h.
   For a tree of height h, this would mean that the number of internal nodes would increase by n and the number of leaves would increase by at least $2n_{h-1}$ and at most $3n_{h-1}$.
   New internal nodes: $m_{h-1} + n_{h-1} = m_h \rightarrow m_{h-1} = m_h - n_{h-1} \rightarrow$
   $m_h - n_{h-1} \leq n_{h-1} - 1 \rightarrow m_h \leq 2n_{h-1} - 1$
   New leaves: $2n_{h-1} \leq n_h \leq 3n_{h-1} \rightarrow 2n_{h-1} - 1 \leq n_h - 1 \leq 3n_{h-1} - 1 \rightarrow$
   $m_h \leq 2n_{h-1}$ -1$\leq n_h$ -1$\leq 3n_{h-1}$ -1$\rightarrow m_h \leq n_h$ -1
   $\therefore m_h \leq n_h$ -1holds for all h ≥ 0.

Dennis Kuzminer
CSCI-UA 310-001 PS1

7.

a. Base case: k = 0
F2 = F1 + F0 → 0 + 1 = 1 → From definition
From sigma notation → 1 + 0 = 1, and 1 = 1, so the claim holds for k = 0.
Inductive step: k+1 holds. Assume that the claim is true. $F_{k+2} = 1 + \sum_{i=0}^{k} F_i$
$F_{k+3} = F_{k+1} + F_{k+2} = 1 + \sum_{i=0}^{k+1} F_i \rightarrow 1 + F_{k+1} + \sum_{i=0}^{k} F_i = F_{k+1} + F_{k+2}$ → which can be simplified to by canceling out $F_{k+1} \rightarrow F_{k+2} = 1 + \sum_{i=0}^{k} F_i$
We can see that the claim holds for k+1. $\therefore F_{k+2} = 1 + \sum_{i=0}^{k} F_i$ holds for all k ≥ 0.

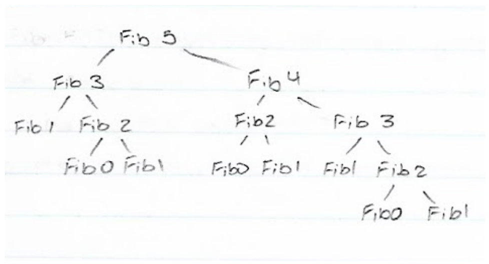
b. Base case 1: k = 0
F2 = F1 + F0 → 0 + 1 = 1 → From definition
ϕ^(0+1) → 1.61803398875 ≥ 1, so the claim holds for k = 0.
Base case 2: k = 1
F3 = F1 + F2 → 1 + 1 = 2 → From definition. ϕ^(1+1) = 2.61803399
Inductive step: k+1 holds. Assume that the claim is true. $F_{k+2} \leq \text{��}^{k+1}$
$F_{k+3} \leq \text{��}^{k+2} \rightarrow F_{k+1} + F_{k+2} \leq \text{��} * \text{��}^{k+1}$ F_{k+2} is at most ��$^{k+1}$ and F_{k+1} is at most ��^{k} → ��^{k} + ��$^{k+1}$ ≤ �� * ���^{k}$^{k+1}$(1 + ��) ≤ ��^{2} (1 * ���^{k}) ≤ ��^{2}
2.61803399≤2.61803399 → We can see that the claim holds for k+1.
$\therefore F_{k+2} \leq \text{��}^{k+1}$

c. Base case 1: k = 0
F2 = F1 + F0 → 0 + 1 = 1 → From definition
ϕ^(0) → 1 ≤ 1, so the claim holds for k = 0.
Base case 2: k = 1
F3 = F1 + F2 → 1 + 1 = 2 → From definition. ϕ^(1) = 1.61803398875
Base case 2: k = 2
F4 = F3 + F2 → 1 + 2 = 3 → From definition. ϕ^(2) = 2.61803399
The claim holds for base case 1 and 2
Inductive step: k+1 holds. Assume that the claim is true. $F_{k+2} \geq \text{��}^{k}$
$F_{k+3} \geq \text{��}^{k+1} \rightarrow F_{k+1} + F_{k+2} \geq \text{��} * \text{��}^{k}$ F_{k+2} is at least ��^{k} and F_{k+1} is at least ��$^{k-1}$ → ��$^{k-1}$ + ��^{k} ≥ ������$^{k-1}$^{k}(1 + ��) ≥ ��^{2} → * (1 + ���$^{k-1}$) ≥ ��^{2}
2.61803399≥2.61803399 → We can see that the claim holds for k+1.
$\therefore F_{k+2} \geq \text{��}^{k}$

Dennis Kuzminer
CSCI-UA 310-001 PS1

8.

a.



b. Base case n = 0

G(n)=2F(n+1)-1→ 1 = 2(1)-1 → 1=1

Inductive step: Assume G(k)=2F(k+1)-1 holds for all n up to k. Prove k+1 also holds: G(k+1)=2F(k+2)-1→

G(k)+G(k-1)+1=2F(k)+2F(k+1)-1→

2F(k-1+1)-1+G(k)+1=2F(k)+2F(k+1)-1→

G(k)+1=2F(k+1) →

G(k)=2F(k+1)-1→

∴G(n)=2F(n+1)-1 holds for all n≥0

c.
```
int fib(int n) {
        int fib[] = new int[n+2];
        for(int i = 0; i <= n; i++) {
                if(i == 0) {
                        fib[i] = 0;
                } else {
                        if(i == 1) {
                                fib[i] = 1;
                        } else {
                                fib[i] = fib[i-1] + fib[i-2];
                        }
                }
        }
        return fib[n];
}
```

Dennis Kuzminer
CSCI-UA 310-001 PS1

9.

a. i=0 → vec.append(1) will call resize(1) which will set newsz to 1. Line 13 was executed 0 times.
i=1 → resize(2) will be called which will have line 13 be executed once as size is not yet updated.
i=2 → resize(3) will be called which will have line 13 be executed twice.
i=3 → resize(4) will be called which will have line 13 be executed three times.
It is clear to see that there is a pattern. vec.append is called n times and for every ith iteration, line 13 is called i times. This means that the formula for the number of times line 13 is called is essentially n * n (n(n-1)/2)(number of loops in the outer for-loop * number of loops in the inner for-loop). This means that the number of times line 13 is executed is **Θ(n^2)**.

b. i=0 → vec.append(1) will call resize(1) which will set newsz to 1. Line 13 was executed 0 times.
i=1 → resize(2) will be called which will have line 13 be executed once.
i=2 → resize(3) will be called which will have line 13 be executed twice.
i=3 → resize(4) will be called which will have line 13 not be executed.
i=4 → resize(5) will be called which will have line 13 be executed four times.
When i is a power of 2, only then will the nested loop execute.
$2^{\lfloor log_2(n) \rfloor}$ → Which is essentially **O(n)**

Outer For Loop i=0	Inner For Loop line 13	You can rewrite this to make it so	Outer For Loop i=0	Redistributed series
1	1	that each ‘i’	1	1
2	2	will have	2	1
3	0	only 1 other	3	1
4	4	execution.	4	1
5	0	This makes	5	1
6	0	O(n)	6	1
7	0		7	1
8	8		8	1