- Homework solutions should be neatly written or typed and turned in through **Gradescope** by 11:59pm on the due date. No late homeworks will be accepted for any reason. You will be able to look at your scanned work before submitting it. Please ensure that your submission is legible (neatly written and not too faint) or your homework may not be graded.

- Students should consult their textbook, class notes, lecture slides, instructors, TAs, and tutors when they need help with homework. Students should not look for answers to homework problems in other texts or sources, including the internet. Only post about graded homework questions on Piazza if you suspect a typo in the assignment, or if you don't understand what the question is asking you to do. Other questions are best addressed in office hours.

- Your assignments in this class will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should always explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

- For questions that require pseudocode, you can follow the same format as the textbook, or you can write pseudocode in your own style, as long as you specify what your notation means. For example, are you using "=" to mean assignment or to check equality? You are welcome to use any algorithm from class as a subroutine in your pseudocode. For example, if you want to sort list A using `InsertionSort`, you can call `InsertionSort(A)` instead of writing out the pseudocode for `InsertionSort`.

There are 5 questions for a total of 55 points.

---

1. (10 points) Let $f(n)$ and $g(n)$ be functions from the nonnegative integers to the positive real numbers. Prove the following transitive property from the definition of big-O:

$$\text{If } f(n) \in O(g(n)) \text{ and } g(n) \in O(h(n)) \text{ then } f(n) \in O(h(n)).$$

Solution:

From the definition of big-O, there exists a positive integer $n_1 \in \mathbb{N}$ and a constant $c_1 \in \mathbb{R}_+$ satisfying

$$f(n) \leq c_1 g(n) \quad \forall n \geq n_1, n \in \mathbb{N}$$

From the definition of big-O, there exists a positive integer $n_2 \in \mathbb{N}$ and a constant $c_2 \in \mathbb{R}_+$ satisfying

$$g(n) \leq c_2 h(n) \quad \forall n \geq n_2, n \in \mathbb{N}$$

Thus, for any $n \in \mathbb{N}, n \geq \max(n_1, n_2)$, we have $f(n) \leq c_1 g(n) \leq c_1 c_2 h(n)$, so for a constant $n_3 = \max(n_1, n_2) \in \mathbb{N}$ and $c_3 = c_1 c_2$, we have

$$f(n) \leq c_3 h(n) \quad \forall n \geq n_3, n \in \mathbb{N}$$

which, by the definition of big-O means that $f(n) \in O(h(n))$.

---

2. <u>State True or False</u>:

   (a) (2 points) $2 \cdot (3^n) \in \Theta(3 \cdot (2^n))$

Answer: False

(b) (2 points) $(n^6 + 2n + 1)^2 \in \Omega((3n^3 + 4n^2)^4)$

Answer: True

(c) (2 points) $\log n \in \Omega((\log n) + n)$

Answer: False

(d) (2 points) $n \log n + n \in O(n \log n)$

Answer: True

(e) (2 points) $\log(n^{10}) \in \Theta(\log(n))$

Answer: True

(f) (2 points) $\sum_{i=1}^{n} i^k \in \Theta(n^{k+1})$

Answer: True

(g) (2 points) $(\log(n))^{\log(n)} \in O(n/\log(n))$

Answer: False

(h) (2 points) $n! \in O(2^n)$

Answer: False

3. Given two lists A of size m and B of length n, our goal is to construct a list of all elements in list A that are also in list B. Consider the following two algorithms to solve this problem.

procedure Search1(List A of size m, List B of size n)
1.   Initialize an empty list L.
2.   SORT list B
3.   **for** each item $a \in A$,
4.   **if** (BinarySearch$(a, B) \neq 0$), **then**
5.   Append a to list L.
6.   **return** L

Note: Assume that the SORT algorithm used in Search1 algorithm above takes time proportional to $k \log k$ on an input list of size k.

procedure Search2(List A of size m, List B of size n)
1.   Initialize an empty list L

> 2.   **for** each item $a \in A$,
> 3.   **if** ($\texttt{LinearSearch}(a, B) \neq 0$), **then**
> 4.   Append a to list L.
> 5.   **return** L

Answer the following questions:

(a) (2 points) Calculate the runtime of $\texttt{Search1}$ in Θ notation, in terms of m and n.

> **Solution:** The first sorting has runtime in $\Theta(n \log n)$. Then for each iteration of the second loop, binary search has runtime in $\Theta(\log n)$, and since this is done m times, the time taken by the for loop is in $\Theta(m \log n)$. So the runtime is finally $\Theta((m + n) \log n)$.

(b) (2 points) Calculate the runtime of $\texttt{Search2}$ in Θ notation, in terms of m and n.

> **Solution:** Each iteration of the for loop has runtime in $\Theta(n)$, and there are m such iterations, so the procedure has runtime in $\Theta(nm)$.

(c) (2 points) When $m \in \Theta(1)$, which algorithm has faster runtime asymptotically?

> **Answer:** $\texttt{Search2}$, since the runtime complexity for it becomes $\Theta(n)$, while that for $\texttt{Search1}$ becomes $\Theta(n \log n)$.

(d) (2 points) When $m \in \Theta(n)$, which algorithm has faster runtime asymptotically?

> **Answer:** $\texttt{Search1}$, since the runtime complexity for it becomes $\Theta(n \log n)$ while that for $\texttt{Search2}$ becomes $\Theta(n^2)$.

(e) (2 points) Find a function $f(n)$ so that when $m \in \Theta(f(n))$, both algorithms have equal runtime asymptotically.

> **Answer:** If $m \in \Theta(\log n)$, then both programs have runtime in $\Theta(n \log n)$.

4. Show that if c is a positive real number, then $g(n) = 1 + c + c^2 + ... + c^n$ is:

(a) (3 points) $\Theta(1)$ if $c < 1$.

> **Solution:** If $c < 1$, we have $1 < g(n) = 1 + c + c^2 + \cdots + c^n < 1 + c + c^2 + \cdots = \frac{1}{1-c}$, so $g(n)$ is bounded between two constants and is hence $\Theta(1)$.

(b) (3 points) $\Theta(n)$ if $c = 1$.

> **Solution:** If $c = 1$, all the terms on the right become 1, so $g(n) = n + 1$. Since for $n > 1$, $n < g(n) < 2n$, we have $g(n) \in \Theta(n)$.

(c) (3 points) $\Theta(c^n)$ if $c > 1$.

> **Solution:** Using the formula for the sum of a geometric progression, we have $c^n < g(n) = \frac{c^{n+1}-1}{c-1} < \frac{c^{n+1}}{c-1} = \left(\frac{c}{c-1}\right)c^n$ for all $n > 1$, so $g(n) \in \Theta(c^n)$.

5. The Fibonacci numbers $F_0, F_1, ...,$ are defined by

$$F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$$

(a) (4 points) Use induction to prove that $F_n \geq 2^{0.5n}$ for $n \geq 6$.

> **Solution:**
> **Base cases:**
>
> 1. $n = 6 : F_6 = 8, 2^3 = 8$
>
> 2. $n = 7 : F_7 = 13, 2^{3.5} < 11.4$
>
> **Inductive hypothesis:** For all k satisfying $6 \leq k \leq n$, we have $F_k \geq 2^{0.5k}$.
> **Inductive step:** We have
>
> $$F_{n+1} = F_n + F_{n-1} \geq 2^{0.5n} + 2^{0.5(n-1)} = 2^{0.5(n+1)} \cdot \left(\frac{1}{\sqrt{2}} + \frac{1}{2}\right) > 2^{0.5(n+1)}$$
>
> Thus, our induction is complete.

(b) (4 points) Use induction to prove that $F_n \leq 2^n$ for $n \geq 0$.

> **Solution:**
> **Base cases:**
>
> 1. $n = 0 : F_0 = 0, 2^0 = 1$
>
> 2. $n = 1 : F_1 = 1, 2^1 = 2$
>
> **Inductive hypothesis:** For all k satisfying $0 \leq k \leq n$, we have $F_k \leq 2^k$.
> **Inductive step:** We have
>
> $$F_{n+1} = F_n + F_{n-1} \leq 2^n + 2^{n-1} = 2^{n+1} \cdot \left(\frac{1}{2} + \frac{1}{4}\right) < 2^{n+1}$$
>
> Thus, our induction is complete.

(c) (2 points) What can we conclude about the growth of F_n?

> **Solution:** This shows that the growth of F_n is exponential in nature.