- 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:

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

If f(n) is O(g(n)), then there are constants $c_o > 0$ and $n_o > 0$ such that $f(n) < c_o$*g(n) for all $n > n_o$.
If g(n) is O(h(n)), then there are constants $c_1 > 0$ and $n_1 > 0$ such that $g(n) < c_1$*h(n) for all $n > n_1$.
So we can write c_o*$g(n) < c_1$*c_o*h(n) for all $n > n_1$
Which implies $f(n) < c_o$*$g(n) < c_1$*c_o*h(n) for all $n > max(n_1, n_o)$
Hence, $f(n) < c_1$*c_o*h(n) for all $n > max(n_1, n_o)$
So we have constants $C = c_o * c_1$ and $N = max(n_1, n_0)$ such that $f(n) < C * h(n)$ for $n > N$.
Hence, from definition of BigO, we can say f(n) = O(h(n)).

2. State True or False:
   (a) (2 points) $2 \cdot (3^n) \in \Theta(3 \cdot (2^n))$ False
   (b) (2 points) $(n^6 + 2n + 1)^2 \in \Omega((3n^3 + 4n^2)^4)$ True
   (c) (2 points) $\log n \in \Omega((\log n) + n)$ True
   (d) (2 points) $n \log n + n \in O(n \log n)$ True
   (e) (2 points) $\log(n^{10}) \in \Theta(\log(n))$ True
   (f) (2 points) $\sum_{i=1}^{n} i^k \in \Theta(n^{k+1})$ False
   (g) (2 points) $(\log(n))^{\log(n)} \in O(n/\log(n))$

(h) (2 points) $n! \in O(2^n)$ 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** (`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 `Search1` in Θ notation, in terms of m and n.
To sort list B, time required is $\Theta(nlogn)$
The best and worst case running complexity of Binary Search in list B is $\Theta(1)$ and $\Theta(logn)$ respectively. Appending to list L takes $\Theta(1)$ time
So repeating binary search for all elements in A, will take time complexity of order $\Theta(mlogn)$.
Hence, the overall algo search1 has complexity $\Theta((n+m)logn)$

(b) (2 points) Calculate the runtime of `Search2` in Θ notation, in terms of m and n.
Linear search for an element in B will have time complexity $\Theta(n)$. and appending to a list takes $\Theta(1)$ time
For m elements in list A, total time complexity will be $\Theta(m * n)$.

(c) (2 points) When $m \in \Theta(1)$, which algorithm has faster runtime asymptotically? search2

(d) (2 points) When $m \in \Theta(n)$, which algorithm has faster runtime asymptotically? search1

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

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 $c < 1$.
if $c < 1, g(n) = 1/(1-c)$

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

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

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 $n \geq 6$.

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

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