Please solve the following 2 problems, both of which have multiple parts.

**Some important homework policies**

- Groups of one or two students may work together. They should submit a single copy of their assignment using one of their eLearning accounts. Everybody in the group will receive the same grade.

- Each group must write their solutions in their own words. Clearly print your name(s), the homework number (Homework 1), and the problem number at the top of every page in case we print anything. Start each numbered homework problem on a new page.

- Unless the problem states otherwise, you must justify (prove) (argue) that your solution is correct.

- Any illegible solutions will be considered incorrect, so you might consider using \LaTeX{} to typeset your solutions. There is a template provided on the course website to help you get started.

- If you use outside sources or write solutions in close collaboration with others outside your group, then you may cite that source or person and still receive full credit for the solution. Material from the lecture, the textbook, lecture notes, or prerequisite courses need not be cited. Failure to cite other sources or failure to provide solutions in your own words, even if quoting a source, is considered an act of academic dishonesty.

- The homework is assigned to give you the opportunity to learn where your understanding is lacking and to practice what is taught in class. Its primary purpose is not for Kyle to grade how well you paid attention in class. Read through the questions early. Do not expect to know the answers right away. Questions are not necessarily given in order of difficulty. *Please, please, please* attend office hours or email Kyle so he can help you better understand the questions and class material. Seriously, Kyle enjoys busy office hours.

- You may assume that any reasonable operation involving a constant number of objects of constant complexity can be done in $O(1)$ time. Clearly state your assumptions if they are not something we already used in lecture.

See https://personal.utdallas.edu/~kyle.fox/courses/cs4349.006.21f/about/ and https://personal.utdallas.edu/~kyle.fox/courses/cs4349.006.21f/writing/ for more detailed policies before you begin. If you have any questions about these policies, please do not hesitate to ask during lecture, in office hours, or through email.
1. (a) Truthfully write the phrase “I have read and understand the policies on the course website.”

(b) Consider the following algorithm: Procedure Study(topics[1 .. n], exams[1 .. t]) gives instructions on how one might study during a single course with n lectures and t exams. Parameter topics[1 .. n] is an array of lecture topics where topics[i] is the topic covered during the i-th lecture of the course. Parameter exams[1 .. t] is an array of distinct integers between 1 and n. For all k between 1 and t, there is a cumulative exam held immediately after lecture exams[k]. The array exams[1 .. t] is given with its entries sorted in ascending order. Also, University regulations limit the total number of exams t to be at most n/4.

\[
\text{Study(topics[1 .. n], exams[1 .. t]):}
\]

\[
\text{nextExam} \leftarrow 1
\]

for i \leftarrow 1 to n

Study topics[i].

if nextExam \leq t

if exams[nextExam] = i

for j \leftarrow 1 to i

Study topics[j].

nextExam \leftarrow nextExam + 1

Forget everything over Winter Break.

Suppose studying any single topic topics[i] takes \( \Theta(1) \) time. Using \( \Theta \)-notation in terms of \( n \) only, give a tight asymptotic bound on the maximum amount of time spent studying while following the instructions given by Study(topics[1 .. n], exams[1 .. t]). You should justify your solution by arguing 1) you cannot spend any more time than you claim (argue for a big-Oh bound) and 2) there is some placement of exams that makes it possible to achieve your claimed study time (argue for a \( \Omega \)-bound).

Advice: For the upper bound argument (and most other times you try to argue a big-Oh bound), you only need to state an upper bound on how many iterations each loop might undertake.

(c) Sort the functions of \( n \) listed below from asymptotically smallest to asymptotically largest, indicating ties if there are any. **Do not submit proofs for this problem.** To simplify your answers, write \( f(n) \ll g(n) \) to mean \( f(n) = o(g(n)) \), write \( f(n) \equiv g(n) \) to mean \( f(n) = \Theta(g(n)) \), and list all the functions in a sequence of these inequalities. For example, if the given functions were \( n^2, n, \binom{n}{2}, \) and \( n^3 \) then the only correct answers would be “\( n \ll n^2 \ll \binom{n}{2} \ll n^3 \)” and “\( n \ll \binom{n}{2} \ll n^2 \ll n^3 \).”

\[
\begin{align*}
2^n & \quad n^2 & \quad n & \quad \log_2 n & \quad \sqrt{n} \\
4^n & \quad \ln^3 n & \quad 17n & \quad n + 500 & \quad 3 - \cos n \\
2^{10 \lg n} & \quad \lg(7n) & \quad 250 & \quad \lg^{0.6} n & \quad n \log n
\end{align*}
\]

Advice: You should be able to solve this problem using only what is written for Lecture 2 along with basic algebraic rules for manipulating logs, polynomials, and exponentials.

2. The \textit{nth Fibonacci binary tree} \( \mathcal{F}_n \) is defined recursively as follows:

- \( \mathcal{F}_1 \) is a single root node with no children.
For all $n \geq 2$, $F_n$ is obtained from $F_{n-1}$ by adding a right child to every leaf and adding a left child to every node that has only one child.

![Figure 1](image.png)

**Figure 1.** The first six Fibonacci binary trees. In each tree $F_n$, the subtree of gray nodes is $F_{n-1}$.

(a) **Using the following template**, prove that the number of leaves in $F_n$ is precisely the $n$th Fibonacci number: $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for all $n \geq 2$. **You must follow the template to receive credit for this part of the problem.**

Let $n$ be an arbitrary non-negative integer. Assume that for any non-negative integer $k < n$, the number of leaves in $F_k$ is precisely $F_k$. There are several cases to consider:

- Suppose $n$ is . . .
- Suppose $n$ is . . .
- . . .
- Suppose $n$ is . . .
  The induction hypothesis implies that . . .

In each case, we conclude the number of leaves in $F_n$ is precisely $F_n$.

(b) How many nodes does $F_n$ have? Give an exact, closed-form answer in terms of Fibonacci numbers. You must prove your answer is correct, but you are free to write the proof as you’d like this time. (We still recommend following the template, though.)