Name:

Student ID #:

- You are expected to work on this assignment on your own
- Use pseudocode, Python-like or English to describe your algorithms. Absolutely no C++/C/Java
- When designing an algorithm, you are allowed to use any algorithm or data structure we explained in class, without giving its details, unless the question specifically requires that you give such details
- Always remember to analyze the time complexity of your algorithms
- Homework has to be submitted electronically on Gradescope by the deadline. No late assignments will be accepted
Problem 1. (10 points)

Using the Master Theorem, give an asymptotic tight bound for $T(n)$ defined by the following recurrence relation

$$T(n) = \begin{cases} 
2 & n = 2 \\
4T\left(\sqrt{n}\right) + \log^2 n & n > 2 
\end{cases}$$

You will need to apply an appropriate substitution to apply the Theorem.

Answer:
Problem 2. (10 points)

We have seen in class that the procedure \textsc{Merge} in the Mergesort algorithm takes two sorted arrays of size \(n\) and produces one fully sorted array of size \(2n\) in \(O(n)\) time. Use the decision tree method to prove a \(2n - o(n)\) lower bound\(^1\) for the problem of merging two sorted arrays, each containing \(n\) items.

Answer:

\(^1\)The little-oh notation is used here to denote an upper bound that is \textbf{not} asymptotically tight. Formally, we say that \(f(n) \in o(g(n))\) if for any positive constant \(c\) we can find a constant \(n_0\) such that \(0 \leq f(n) < cg(n)\) for all \(n \geq n_0\).
Problem 3. (10 points)

Show how to implement a queue using two stacks $S_1$ and $S_2$ so that the amortized cost of each operation on the queue is $O(1)$.

1. Give the pseudocode for the `ENQUEUE(x)` operation and the `DEQUEUE()` operation (you can omit error checking for underflow and overflow of the stacks).

2. Use the accounting method to charge each operation a constant amortized cost and prove that a sequence of $n$ `ENQUEUE` and `DEQUEUE` cost $O(n)$ time overall.

Answer: