This exam is worth 30 points. It consists of four problems, each worth 10 points. The sum of the three highest scored problems defines the final grade.

If you want to ask a question, write it on the provided sheet of paper and raise your hand. I will collect it, write down the answer and return it back to you.

If you need more scrap paper, raise your hand. Use only the scrap paper provided.

Please turn off your cell-phones and other electronic devices.
Problem 1

Rank the following functions by order of growth; that is, find an arrangement \(g_1(n), g_2(n), \ldots, g_{10}(n)\) of functions satisfying \(g_i(n) = \Omega(g_{i+1}(n))\) for every \(i \in \{1, \ldots, 9\}\). Partition your list into equivalence classes such that \(f(n)\) and \(g(n)\) are in the same class if and only if \(f(n) = \Theta(g(n))\). You do not have to prove your answers.

\[
\begin{array}{cccc}
  n \log_3 n & n^{5.3} + 2n^7 & n \log(n^2) & 3^n & 5\sqrt{n} \\
  \frac{1}{10000} n^7 & 1 & n^{-3} & 2^{7 \log n} & 1 + 4n^{-3}
\end{array}
\]

Recall that in this class we use \(\log n\) to denote the logarithm base 2.
Problem 2

Consider the following divide-and-conquer algorithm that takes as input an array $A = [a_1, a_2, \ldots, a_k]$, and two indices $\ell$ and $r$ such that $1 \leq \ell \leq r \leq k$:

```
WhatAmIDoing(A, \ell, r)
1. If $\ell = r$ then return $a_\ell$.
2. Else
3. Let $m = \lfloor (\ell + r)/2 \rfloor$.
4. Let $x = $ WhatAmIDoing$(A, \ell, m)$.
5. Let $y = $ WhatAmIDoing$(A, m + 1, r)$.
6. Return $min\{x, y\}$.
```

- Trace the algorithm for the following input: $A = [2, 5, 7, 3, 4, 1, 6, 8]$, and $\ell = 1$, $r = 8$. More precisely, list every recursive call (state the values of its parameters $\ell$ and $r$), and for each recursive call state its return value:

- What is this algorithm doing? Describe in words the value returned by the algorithm:

- Analyze the running time of the algorithm. More precisely, let $T(n)$ be the running time of the algorithm, where $n = r - \ell + 1$. For simplicity, suppose that $n$ is a power of two.

  (a) State the recurrence for $T(n)$ and explain where each of the terms in the recurrence comes from:

  (b) Solve the recurrence for $T(n)$, i.e., find a function $g(n)$ such that $T(n) = O(g(n))$ (ideally, $T(n) = \Theta(g(n))$).
Problem 3

Given is a sequence of numbers $a_1, a_2, \ldots, a_n$ and a number $b$. We want to know whether there are two numbers $a_i$ and $a_j$, $i \neq j$, such that $a_i + a_j = b$. Give an $O(n \log n)$ algorithm that determines whether such $i$ and $j$ exist. Reason the running time of your algorithm.
Problem 4

Recall the stable matching problem, where there are \( n \) men and \( n \) women and each has a preference list for the opposite gender. The goal is to find a stable matching, where every man is paired with exactly one woman and there is no potential couple that would prefer to be matched over their current choice of partners. More precisely, a matching is stable if there is no potential couple \((m, w)\) such that \( m \) prefers \( w \) to the woman \( w' \) he is currently paired with and \( w \) prefers \( m \) to the man \( m' \) she is currently paired with.

Consider the following greedy algorithms. For each algorithm, determine whether it produces a stable matching or not.

• Suppose the men are named 1, 2, \ldots, \( n \). Loop through the men from \( m = 1 \) to \( n \). For man \( m \), go through his preference list until you find a woman \( w \) who does not have a partner yet. Pair man \( m \) with woman \( w \).

Does the algorithm work? If yes, reason its correctness. If not, provide a counterexample.

• While there are unassigned people, do the following. Let \( M \) be the set of men who are currently without partners. Let \( k = 1 \). Keep increasing \( k \) until you find a man \( m \in M \) such that the woman at the \( k \)-th place of his preference list is available. Let \( w \) be the woman. If \( w \) appears at the \( k \)-th place for other men in \( M \), take \( m \) to be the man who appears the highest on her preference list. Pair \( m \) with \( w \).

Does the algorithm work? If yes, reason its correctness. If not, provide a counterexample.

• For both algorithms, estimate their running times. State the running times in asymptotic notation and reason your estimates.