Final Exam

- Do not open this booklet until you are directed to do so. Read all the instructions on this page.
- When the exam begins, write your name on every page of this booklet.
- This exam contains five problems, some with multiple parts. You have 180 minutes to earn 180 points.
- This booklet contains 17 pages, including this one. Two extra sheets of scratch paper are attached. Please detach them before turning in your exam at the end of the examination period.
- This exam is closed book. You may use two handwritten A4 or $8 \frac{1}{2}'' \times 11''$ crib sheets. No calculators or programmable devices are permitted.
- Write your solutions in the space provided. If you need more space, write on the back of the sheet containing the problem. Do not put part of the answer to one problem on the back of the sheet for another problem, since the pages may be separated for grading.
- Do not waste time and paper rederiving facts that we have studied. It is sufficient to cite known results.
- Do not spend too much time on any one problem. Read them all through first, and attack them in the order that allows you to make the most progress.
- Show your work, as partial credit will be given. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it. Be neat.
- Good luck!

<table>
<thead>
<tr>
<th>Problem</th>
<th>Parts</th>
<th>Points</th>
<th>Grade</th>
<th>Grader</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16</td>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>180</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Name: ________________________________

Circle the name of your recitation instructor:

Enoch  Jim  Sergey  Shivani
Problem 1. True or False, and Justify [80 points] (16 parts)

Circle T or F for each of the following statements to indicate whether the statement is true or false, respectively. If the statement is correct, briefly state why. If the statement is wrong, explain why. The more content you provide in your justification, the higher your grade, but be brief. Your justification is worth more points than your true-or-false designation.

T F For every two positive functions $f$ and $g$, if $g(n) = O(n)$, then $f(g(n)) = O(f(n))$.

T F Suppose $f(n) = 4f(n/4) + n$ for $n > 8$, and $f(n) = O(1)$ for $n \leq 8$. Similarly, suppose $g(n) = 3g(n/4) + n\log n$ for $n > 8$, and $g(n) = O(1)$ for $n \leq 8$. Then $f(n) = \Theta(g(n))$. 
T F Suppose that a randomized algorithm $A$ has expected running time $\Theta(n^2)$ on any input of size $n$. Then it is possible for some execution of $A$ to take $\Omega(2^n)$ time.

T F Suppose we maintain a hash table with $m$ slots using chaining and a hash function chosen from a universal hash family. If we insert $n > m$ keys into this (initially empty) hash table, then the total number of collisions is $O(n/m)$ in expectation. (Recall that a collision is a pair of distinct keys that hash to the same slot.)
T F  Building an $n$-element heap requires $\Theta(n \lg n)$ time.

T F  Given an unsorted array $A$ of $n$ integers, let $x_i$ denote the $2^i$th smallest element in $A$. Then we can compute $\sum_{i=0}^{\lfloor \lg n \rfloor} x_i$ in $O(n)$ time.
T  F  Suppose that you have a 2-3 tree $T_1$ and AA-tree (from problem set 4) $T_2$, each storing the same set of keys. Then in-order traversals of $T_1$ and $T_2$ can result in different sequences of keys.

T  F  There are at least two distinct 2-3 trees containing keys 1, 2, 3, 4, 5.
T F Graduating from MIT requires passing $n$ specified classes. You decide to take each class every semester until you pass it. Suppose that, every semester you take a class, you have a 50% chance of passing it and a 50% chance of having to drop it. Then you will graduate in $O(\lg n)$ semesters with high probability.
T F Suppose that you have two deterministic online algorithms, $A_1$ and $A_2$, with competitive ratios $c_1$ and $c_2$, respectively. Consider the randomized online algorithm $A^*$ that flips a fair coin once at the beginning; if the coin comes up heads, it runs $A_1$ from then on; if the coin comes up tails, it runs $A_2$ from then on. Then the expected competitive ratio of $A^*$ is at least $\min\{c_1, c_2\}$.

T F In a connected undirected graph $G = (V, E, w)$ with nonnegative edge weights, the shortest-path tree from any source vertex $s \in V$ is a minimum spanning tree of $G$. (Recall that the shortest-path tree from $s$ consists of the edges $\{(\pi(v), v) : v \in V - \{s\}\}$ where, for each $v \in V - \{s\}$, $(\pi(v), v)$ is the last relaxed incoming edge at $v$ in an execution of Dijkstra’s single-source shortest-paths algorithm from $s$.)

**T F**  Reweighting a graph with negative edge weights but no negative-weight cycles, as in Johnson’s algorithm, can be used to solve the single-source shortest-paths problem more efficiently than Bellman-Ford.

**T F**  Every problem in NP can be solved in exponential time.
T F For any decision problem $\pi$ in NP, define the input size $n$ as the parameter $k$ to fix. Then $\pi$ is fixed-parameter tractable with respect to $n$.

T F Define an **independent set** of a graph $G = (V, E)$ to be a subset $S \subseteq V$ of vertices such that $V - S$ is a vertex cover of $G$. Every 2-approximation algorithm for finding a minimum vertex cover is also a 2-approximation algorithm for finding a maximum independent set.
Problem 2. One, One Room; Two, Two Rooms; Ah Ha Ha! [20 points] (3 parts)

You are maintaining a hotel room reservation system for an \( n \)-day period, with dates labeled \( 1, 2, \ldots, n \). Your reservation system must support two operations:

- \( \text{RESERVE}(i, j) \) makes a room reservation for the dates \( i, i+1, \ldots, j \).
- \( \text{COUNT}(i) \) computes how many rooms are currently reserved on day \( i \).

Your goal is to construct a data structure that supports both operations in \( O(\log n) \) time. Assume that \( n \) is an exact power of two.

You decide to maintain your data in the form of a perfectly balanced binary tree. The root corresponds to the entire interval \([1..n]\); the root’s left child corresponds to the interval \([1..n/2]\); the root’s right child corresponds to the interval \([(n/2+1)\ldots n]\); etc. At the (bottom) leaf level, each leaf corresponds to a single day, and the leaf order matches the day order. Thus there are exactly \( n \) leaves and \( 1 + \lfloor \log n \rfloor \) levels.

(a) What additional information would you maintain in the nodes in order to support the updates and queries efficiently?
(b) Briefly describe how you would implement the $\text{COUNT}(i)$ operation. Briefly justify why the running time is $O(\lg n)$.

(c) Briefly describe how you would implement the $\text{RESERVE}(i, j)$ operation. Briefly justify why the running time is $O(\lg n)$. 
Problem 3. **Forty Two** [30 points] (4 parts)

Professor Hackermann has finally cracked the meaning of life, the universe, and everything: it is $H(42, 42)$ where the function $H(m, n)$ is defined by the following unusual recurrence:

\[
H(m, 1) = m^2; \\
H(1, n) = n^3; \\
H(m, n) = H\left(FOO(m, n), n\right) + H\left(m, BAR(m, n)\right) \quad \text{for all other values of } m, n \geq 1.
\]

Professor Hackermann knows how to compute $FOO(m, n)$ and $BAR(m, n)$ in $O(1)$ time for given values of $m, n \geq 1$. The catch is that $FOO(m, n)$ and $BAR(m, n)$ can sometimes be larger than $m$ and $n$, so the recurrence does not obviously terminate. Nonetheless, both $FOO(m, n)$ and $BAR(m, n)$ have value 1 often enough that the recursive formula may allow computing $H(m, n)$.

**(a)** The professor hires you to compute $H(3, 4)$ by hand, using the following information about $FOO$ and $BAR$:

\[
\begin{align*}
FOO(3, 4) &= 7; & BAR(3, 4) &= 1; \\
FOO(7, 4) &= 10; & BAR(7, 4) &= 2; \\
FOO(10, 4) &= 1; & BAR(10, 4) &= 1; \\
FOO(7, 2) &= 1; & BAR(7, 2) &= 1.
\end{align*}
\]

Show your work.
To understand whether $H(42, 42)$ can be computed with the recursive formula, Professor Hackermann sets out to understand which pairs $(m', n')$ arise from the recursion. The professor defines $\text{descendants}(m, n)$ to be the set of pairs $(m', n')$ for which $H(m', n')$ is required to compute $H(m, n)$, i.e.,

$$\text{descendants}(m, n) = \left\{ \left( \text{FOO}(m, n), n \right), \left( m, \text{BAR}(m, n) \right) \right\} \cup \text{descendants}(\text{FOO}(m, n), n) \cup \text{descendants}(m, \text{BAR}(m, n)).$$

Note that $\text{descendants}(m, n)$ does not necessarily include $(m, n)$.

**(b)** To thwart critics who claim that pairs $(m', n')$ in $\text{descendants}(m, n)$ can grow without bound, Professor Hackermann makes the following conjecture:

**Conjecture 1** For every $m, n \geq 1$, and for every pair $(m', n') \in \text{descendants}(m, n)$, we have both $m' \leq (m + n)^3$ and $n' \leq (m + n)^3$.

Give an algorithm that, on input $m, n \geq 1$, determines whether Conjecture 1 is true for this pair of integers, i.e., whether $m', n' \leq (m + n)^3$ for every pair $(m', n') \in \text{descendants}(m, n)$. Your algorithm must run in time polynomial in $m + n$. 
(c) More critics claim that the professor’s recursive formula is useless because it could be cyclic: the computation of $H(m, n)$ could require the computation of $H(m, n)$ itself, leading to an infinite recursion. To thwart these critics, Professor Hackermann makes another conjecture:

**Conjecture 2** For every $m, n \geq 1$, we have $(m, n) \notin \text{descendants}(m, n)$.

Give an algorithm that, on input $m, n \geq 1$, determines whether Conjecture 2 is true for this pair of integers, i.e., whether $(m, n) \notin \text{descendants}(m, n)$. Your algorithm must run in time polynomial in $m + n$, and it may assume that Conjecture 1 holds.
(d) Assuming Conjectures 1 and 2 hold, give an algorithm to compute $H(m, n)$ with running time polynomial in $m + n$. What is the asymptotic running time of your algorithm?
Problem 4. Cliquish Behavior [20 points] (4 parts)

Prof. Vernon has come up with the following divide-and-conquer algorithm, BREAKFAST, for finding a clique\(^1\) in an undirected graph \(G = (V, E)\):

1. Number the vertices in \(V\) as \(1, 2, \ldots, n\), where \(n = |V|\).
2. If \(n = 1\), return \(V\).
3. Partition the vertices into the two sets \(V_1 = \{1, 2, \ldots, \lfloor n/2 \rfloor\}\) and \(V_2 = \{\lceil n/2 \rceil + 1, \ldots, n\}\).
4. Let \(G_1\) be the subgraph of \(G\) induced by \(V_1\), and similarly let \(G_2\) be the subgraph of \(G\) induced by \(V_2\). (In other words, the edges of \(G_1\) are all edges of \(G\) that connect pairs of vertices in \(V_1\), and the edges of \(G_2\) are those of \(G\) that connect pairs of vertices in \(V_2\).)
5. Recursively find cliques \(C_1 = \text{BREAKFAST}(G_1)\) and \(C_2 = \text{BREAKFAST}(G_2)\).
6. Combine these two cliques as follows:
   - Initialize \(C_1^+ \leftarrow C_1\) and \(C_2^+ \leftarrow C_2\).
   - For every vertex \(v \in C_2\), if \(v\) is adjacent to every vertex of \(C_1^+\), then add \(v\) to \(C_1^+\).
   - For every vertex \(u \in C_1\), if \(u\) is adjacent to every vertex of \(C_2^+\), then add \(u\) to \(C_2^+\).
   - Return the larger of \(C_1^+\) and \(C_2^+\).

(a) Briefly argue that the BREAKFAST algorithm always returns a clique of \(G\).

(b) Give an asymptotically tight upper bound on the running time of the BREAKFAST algorithm.

---

\(^1\)For a graph \(G\), a clique \(C \subset V\) is a subset of vertices that are all interconnected by edges.
(c) Give an example of a graph $G$ where the algorithm produces a clique of less than maximum size.

(d) If the professor could modify algorithm BREAKFAST so as to find the largest clique without increasing the asymptotic running time, what would this tell you about the classes P and NP? Briefly explain your answer.
SCRATCH PAPER — Please detach this page before handing in your exam.