Practice Exam 1

- Do not open this exam 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 exam booklet.

- This exam contains 6 problems, some with multiple parts. You have 120 minutes to earn 105 points.

- This exam booklet contains 10 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 one handwritten 8½ × 11 or A4 crib sheet. 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!

Name: __________________________ ID: __________

Section you want your test returned in (circle one): 001 002

<table>
<thead>
<tr>
<th>Problem</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Points</td>
<td>12</td>
<td>18</td>
<td>20</td>
<td>15</td>
<td>15</td>
<td>25</td>
<td>105</td>
</tr>
<tr>
<td>Grade</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problem 1 (Big-$O$ Notation, 12 points). Rank the following functions by increasing order of growth, that is, find an arrangement $g_1, ..., g_{12}$ of the functions satisfying $g_1(n) = O(g_2(n)), g_2(n) = O(g_3(n)), ...$. Break the functions into classes so that $f$ and $g$ are in the same class if and only if $f(n) = \Theta(g(n))$. Note that $\log(\cdot)$ is the base 2 logarithm and $\log_b(\cdot)$ is the base $b$ logarithm.

$$\sum_{i=1}^{3n} (2i + 1), \log_3(n^2), 2^n, n^\frac{1}{\pi^2}, n^{465},$$

$$n \log n, 3^\log n, n^{\log n}, \log(n!), n!, n^n, n^{\log_2 3}$$
Problem 2 (Recurrences, 18 points). Solve the recurrences in parts (a) to (d), expressing your answers using $\Theta$-notation. Whenever possible, apply the Master Theorem and state which case you used. If the Master Theorem does not apply, (i) draw a recursion tree, (ii) specify its height, (iii) estimate the sum of the nodes at each level, and (iv) give the solution to the recurrence.

Assume that $T(n) = \Theta(1)$ for small $n$, and that (when applicable) the Regularity Condition is met.

(a) $T(n) = 3T(n/3) + \sqrt{n}$

(b) $T(n) = 4T(n/2) + 3n^2$

(c) $T(n) = T(n^{1/2}) + \log \log n$

(Continued on next page.)
(d) \( T(n) = 4T(n/5) + n^2 \)

Use the substitution method to prove \( T(n) = \Omega(n^2) \) for \( T(n) \) satisfying:

(e) \( T(n) = T(n/2) + T(4n/5) + T(7n/10) + n \)
Problem 3 (True or False, and Justify, 20 points). 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 better your argument, the higher your grade, but be brief. **No points will be given even for a correct solution if no justification is presented.**

T  F  For all asymptotically nonnegative functions \( f \), \( f(n) + o(f(n)) = \Theta(f(n)) \).

T  F  Insertion Sort takes \( O(n) \) time on the following input of length \( n \):

\[
\left\lfloor \frac{n}{2} \right\rfloor + 1, \left\lfloor \frac{n}{2} \right\rfloor + 2, \ldots, n - 1, n, 1, 2, \ldots, \left\lfloor \frac{n}{2} \right\rfloor - 1, \left\lfloor \frac{n}{2} \right\rfloor.
\]

T  F  The recursion tree for Mergesort, on an array of size \( n \), has \( n \log n \) leaves.

T — F  The smallest element in a max heap can reside in any leaf.

T — F  One can find the successor of a node in a binary search tree in time \( O(1) \) in the worst case.
Problem 4 (Short Algorithms, 15 points). For each of the following algorithm problems, give pseudocode as well as a brief (one or two sentence) justification of correctness.

(a) Give an algorithm that returns the second-largest element in a heap. Your algorithm should run in constant time.

(b) Give an algorithm that computes the sum of all the keys in a binary search tree. Your algorithm should run in time $O(n)$, where $n$ is the number of nodes in the tree.

(c) Give an algorithm that takes an unsorted array of integers and determines if there is any number that appears twice in the array. Your code should run in time $O(n \log n)$ (note: faster is ok!), where $n$ is the length of the array.
Problem 5 (Loop invariants, 15 points). Consider the following problem (actually discussed in CLRS Chapter 4.1):

You are consulting for a small investment company. They give you a price of Google’s shares for the last \( n \) days. Let \( p(i) \) represent the price for day \( i \). During this time period, the company wanted to buy 1,000 shares on some day and sell all these shares on some later day. The company wants to know when they should have bought and when they should have sold the shares in order to maximize the profit. If there was no way to make money during the \( n \) days, you should report this instead.

For example, suppose \( n = 3, p(1) = 9, p(2) = 1, p(3) = 5 \). Then you should return "buy on day 2, sell on day 3".

You mention the problem to Professor Onepass, and she suggests the following algorithm:

\[
\text{BESTTWO DAYS}(p) \\
\quad \triangleright \text{Prices are given in the array } p \\
1 \quad n \leftarrow \text{length}(p) \\
2 \quad \text{if } n = 1 \\
3 \quad \quad \text{then return "No way to make money."} \\
4 \quad buy \leftarrow 1 \\
5 \quad sell \leftarrow 2 \\
6 \quad \text{if } p[1] < p[2] \\
7 \quad \quad \text{then minsofar} \leftarrow 1 \\
8 \quad \quad \text{else } minsofar \leftarrow 2 \\
9 \quad \text{for } k \leftarrow 3 \text{ to } n \\
10 \quad \quad \text{do if } (p[k] - p[minsofar]) > (p[sell] - p[buy]) \\
11 \quad \quad \quad \text{then sell} \leftarrow k \\
12 \quad \quad \text{buy} \leftarrow minsofar \\
13 \quad \quad \text{if } p[k] < p[minsofar] \\
14 \quad \quad \quad \text{then minsofar} \leftarrow k \\
15 \quad \quad \text{if } p[buy] \geq p[sell] \\
16 \quad \quad \quad \text{then return "No way to make money."} \\
17 \quad \text{else return "Buy on day } buy \text{, sell on day } sell"}
\]

(a) Give the running time of the code using asymptotic notation.

(b) State a loop invariant for the \textbf{for} loop in lines 7–12.
(e) Prove that the algorithm is correct using your loop invariant.

(i) Initialization

(ii) Maintenance

(iii) Termination
Problem 6 (Divide and Conquer, 25 points). An array $A[1..n]$ is **unimodal** if it consists of an increasing sequence followed by a decreasing sequence, or more precisely, if there is an index $m \in \{1, 2, ..., n\}$ such that

- $A[i] < A[i + 1]$ for all $i < m$, and
- $A[i] > A[i + 1]$ for all $i \geq m$

In particular, $A[m]$ is the maximum element, and it is the unique element surrounded by smaller elements ($A[m - 1]$ and $A[m + 1]$). For example, the array 3,7,9,10,15,6,5,4,1 is unimodal.

(a) Give a divide-and-conquer algorithm to compute the maximum element of a unimodal input array $A[1..n]$ in $O(\log n)$ time. First, explain your algorithm concisely in English. Second, specify your algorithm using pseudocode.
(b) Explain in a couple of sentences why your algorithm is correct.

(c) Give a recurrence for the worst case running time of your algorithm and solve it.
Problem 4 (Dynamic Programming, 20 points). Let $G = (V, E)$ be an undirected graph with $n$ nodes. A subset of the nodes is called an independent set if no two of them are joined by an edge.

We call a graph $G$ a chain if its nodes can be written as $v_1, v_2, v_3, \ldots, v_n$, with an edge between $v_i$ and $v_j$ if and only if $i$ and $j$ differ by exactly 1. We associate a positive integer weight $w_i$ with each node $v_i$. Our goal is to solve the Chain-IS problem: given a chain $G$, find an independent set whose total weight is as large as possible.

For example, in the following graph, each vertex is marked with its weight. The maximum-weight independent set is colored gray.

(a) Give an example to show that the following “heaviest-first” greedy algorithm does not always find an independent set of maximum total weight:

```
HEAVIEST-FIRST(G)
1 Start with $S$ equal to the empty set
2 while some node remains in $G$
3 do
4 Pick a node $v_i$ of maximum weight in $G$
5 Add $v_i$ to $S$
6 Delete $v_i$ and its neighbors from $G$
7 return $S$
```
(b) Give a dynamic programming algorithm for the Chain-IS problem. For full credit, your algorithm should run in time $O(n)$.

Algorithm:

Explanation of correctness:

Running time analysis:

— End of the exam. Have a great holiday! —