NAME:______________________________

<table>
<thead>
<tr>
<th>Problem 1</th>
<th>Max 10</th>
<th>Obtained</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 (a)</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>2 (b)</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3 (a)</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3 (b)</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>5 (a)</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5 (b)</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5 (c)</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

Comments:

- You can use any of the algorithms covered in class without describing them.
- When describing an algorithm, remember to include an argument for both correctness and running time.
[10 points] Problem 1:

1. The summation \( \sum_{i=0}^{\lg n} \left( \frac{1}{2} \right)^i \) is \( \Theta( ) \).

2. Is it true that \( \sqrt{n} = O(2^{\log_2 n}) \) ?

3. The best case running time of Quicksort is \( \Theta( ) \).

4. Given a heap with \( n \) elements, is it true that you can search for an element in \( O(\log n) \) time?

5. Assume you have \( n \) positive integers in the range 1 through \( k \). Counting Sort sorts the \( n \) integers in \( O( ) \) time using \( O( ) \) additional space.
[20 points ] Problem 2:

a) Using the iteration method find an asymptotic tight bound for the recurrence:

\[
T(n) = \begin{cases} 
1 & \text{if } n \leq 3 \\
T(\sqrt{n}) + 1 & \text{if } n \geq 4 
\end{cases}
\]
b) Show using the substitution method (induction) that the recurrence above has solution
\[ T(n) = O(\lg \lg n). \]
Problem 3:

Let $A$ be an array of $n$ (not necessarily distinct) integers.

a) Describe an $O(n)$-algorithm to test whether any item occurs more than $\lceil n/2 \rceil$ times in $A$.

b) Describe an $O(n)$-algorithm to test whether any item occurs more than $\lceil n/4 \rceil$ times in $A$. 
Problem 4:

In this problem we consider a monotonically decreasing function $f : N \rightarrow Z$ (that is, a function defined on the natural numbers taking integer values, such that $f(i) > f(i + 1)$). Assuming we can evaluate $f$ at any $i$ in constant time, we want to find $n = \min\{i \in N | f(i) \leq 0\}$ (that is, we want to find the value where $f$ becomes negative).

We can obviously solve the problem in $O(n)$ time by evaluating $f(1), f(2), f(3), \ldots f(n)$. Describe an $O(\log n)$ algorithm.

(Hint: Evaluate $f$ on $O(\log n)$ carefully chosen values between 1 and $2n$ - but remember that you do not know $n$ initially).
Problem 5:
The maximum partial sum problem (MPS) is defined as follows. Given an array $A[1..n]$ of integers, find values of $i$ and $j$ with $1 \leq i \leq j \leq n$ such that

$$\sum_{k=i}^{j} A[k]$$

is maximized.

Example: For the array [4,-5,6,7,8,-10,5], the solution to MPS is $i = 3$ and $j = 5$ (sum 21).

To help us design an efficient algorithm for the maximum partial sum problem, we consider the left position $\ell$ maximal partial sum problem ($LMPS_\ell$). This problem consists of finding value $j$ with $\ell \leq j \leq n$ such that

$$\sum_{k=\ell}^{j} A[k]$$

is maximized. Similarly, the right position $r$ maximal partial sum problem ($RMPS_r$), consists of finding value $i$ with $1 \leq i \leq r$ such that

$$\sum_{k=i}^{r} A[k]$$

is maximized.

Example: For the array [4,-5,6,7,8,-10,5] the solution to e.g. $LMPS_4$ is $j = 5$ (sum 15) and the solution to $RMPS_7$ is $i = 3$ (sum 16).
a) Describe $O(n)$ time algorithms for solving $LMPS_\ell$ and $RMPS_r$ for given $\ell$ and $r$. 
b) Using an $O(n)$ time algorithm for $LMPS_i$, describe a simple $O(n^2)$ algorithm for solving $MPS$. 
c) Using $O(n)$ time algorithms for $LMPS_l$ and $RMPS_r$, describe an $O(n \log n)$ divide-and-conquer algorithm for solving $MPS$. 