CS 462: Midterm Exam

Problem 0 (5 points): Write your name on this cover sheet and sign your name to acknowledge that you used no reference materials except for the textbook, the instructor’s notes, and your own notes to complete this exam.

Please write all answers clearly and succinctly. Use the back of the page if you need more room to answer a question.
DO ALL OF PROBLEMS 1-4

Problem 1 (15 points): Give a tight asymptotic bound on each of the following recurrences.

(a) $T(0) = T(1) = 1, T(n) = 4T(\lfloor \frac{n}{2} \rfloor) + 2n^2 + n$ for $n > 1$.

(b) $T(0) = T(1) = 3, T(n) = 4T(\lfloor \frac{n}{2} \rfloor) + 5T(\lfloor \frac{n}{6} \rfloor) + n^2$ for $n > 1$.

(c) $T(0) = T(1) = 2, T(n) = T(\lfloor \frac{n}{4} \rfloor) + T(\lfloor \frac{n}{7} \rfloor) + n$ for $n > 1$. 
Problem 2 (10 points): Prove or disprove: all leaves of a heap are interchangable (that is, the values in any two leaves can be exchanged without violating the heap order property).
**Problem 3 (10 points):** For each of the following sets of inputs, say which sorting algorithm you would use to sort the input. Explain your answers.

1. $10^6$ current students to be sorted by class year (with ties within class year broken arbitrarily)

2. $10^6$ real numbers uniformly distributed over the interval $(-10, 10)$.

3. $10^6$ UPC codes (strings of 12 digits)

4. $10^6$ integers in an unspecified range
Problem 4 (10 points): Consider the problem of finding the fastest way to get from one point to another on a grid of city streets. We can model this problem as moving on an $n$-by-$n$ checkerboard. At any point, we can move up, down, left, or right (as long as such a move is not off the board), and every move has a cost that corresponds to the time it takes to move one block in the city. We want to find the least cost to move from one corner of the board to the opposite corner.

Consider the following attempt to solve the problem:

let $c[x]$ be the best time to get to square $x$

let $t(x, y)$ be the time it takes to get from square $x$ to square $y$

initialize $c[x]$ to 0 for $x =$ the upper left corner

for each row from 1 to $n$
  for each column from 1 to $n$
    find the minimum value of $c[y] + t(y, x)$ for all squares $y$ adjacent to $x$
    let $c[x]$ be that minimum value

output the value for the lower right corner

Will this algorithm work? Justify your answer with a proof or a counterexample.
Problem 5 (15 points):

(a) Draw the min-heap that results from running the BUILD-HEAP algorithm on an array that initially contains \{7, 13, 18, 6, 17, 11, 5, 10, 1\}. Show intermediate results for partial credit.

(b) Show the result of the EXTRACT-MIN operation on the heap from part (a).

(c) Show the result of the changing the 6 to 20 in the heap from part (b) using HEAP-INCREASE-KEY.
Problem 6 (15 points): Illustrate the operation of QUICKSORT on an array that initially contains \{ 4, 9, 10, 7, 2, 3, 1 \}. Part of your answer should include the recursion tree.
**Problem 7 (15 points):** Let $A_1, A_2, A_3, A_4, A_5$ be matrices with respective dimensions $10 \times 10, 10 \times 20, 20 \times 10, 10 \times 5,$ and $5 \times 20$. Find the order in which to compute the product $A_1 A_2 A_3 A_4 A_5$ that minimizes the number of scalar multiplications done.
Problem 8 (10 points): Write the loop invariant for the loop in the code given below, which, given an array of integers in sorted order, finds the frequency of the most frequent value in the array (so if the input is \{1, 1, 2, 2, 2, 3, 4, 4\} the output would be 4 since there are 4 twos and there are more twos than any other number). Your loop invariant should be sufficient to use in a proof that the code is correct (that is, \(i < n\) is not a good answer even though it is always true inside the loop).

```java
highCount = -1;
lastSame = 0;

for (int i = 1; i < n; i++)
{
    if (A[i] != A[i - 1])
    {
        if (i - lastSame > highCount)
            highCount = i - lastSame;
        lastSame = i;
    }
}

if (n - lastSame > highCount)
    highCount = n - lastSame;

return highCount;
```
Problem 9 (10 points): Define \( n(k) \) by the following recurrence relation: \( n(0) = n(1) = n(2) = 1 \), and for \( k > 2 \),

\[
n(k) = \sum_{i=1}^{k-1} n(i) \cdot n(k - i).
\]

Give an efficient algorithm that computes \( n(k) \). Determine the asymptotic efficiency of your algorithm.
Problem 10 (10 points): Does bucket sort still work in expected linear time if there are $\lfloor \frac{n}{2} \rfloor$ buckets instead of $n$ buckets? Justify your answer mathematically.
**Problem 11 (10 points):** The HAS-MAJORITY problem is the problem of, given $n$ integer inputs, determining whether there is one value that occurs more than $n/2$ times (for example, if the inputs are 1 3 5 1 2 then the answer would be “no”, but if the inputs were 2 2 4 5 2 7 8 2 2 then the answer would be yes). Find a linear time algorithm that solves HAS-MAJORITY. Prove that your answer is correct.