CS 242: Data Structures and Algorithms

First Examination (March 7th, Tuesday)

Name: _____ *Total Points: 150*

- *You have 110 minutes to attempt these problems.*
- *Do not spend too much time on any problem. Read through them first and attempt 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. Except for the first problem, 10% of the grade will be reserved for clarity.*

Drill

1. (30 points) Indicate, for each pair of expressions (A, B) in the table below, whether A is O, Θ, Ω, o of B. Assume that n is a variable. Your answer should be in the form of the table with "yes" or "no" written in each box.

A	B	A is $O(B)$	A is $\Theta(B)$	A is $\Omega(B)$	A is $o(B)$
$3^{\log_9 n}$	$\sqrt{n}$				
$\lg^3 n$	$\sqrt{n} \lg n$				
$2^{n^{1/3}}$	$2^{n/3}$				

2. (15 points) Use the master method to provide asymptotically tight solution to the following recurrences. You may assume, if needed, that the regularity condition holds.

$$T(n) = 7T(n/4) + n^2 \quad (1)$$

$$T(n) = 8T(n/4) + n^2 \quad (2)$$

$$T(n) = 9T(n/4) + n^2 \quad (3)$$

3. (30 points) Consider the following method written in Java. Suppose that we invoke it as `TowersOfHanoi(n,1,2,3)`. Write down a recurrence equation for the run time of the method. Solve the recurrence equation.

```
public static void TowersOfHanoi (int n, int fromPeg, int middlePeg, int toPeg
{
    if (n < 0)
        throw new IllegalArgumentException();
    if (n == 0)
        return;

    TowersOfHanoi(n-1, fromPeg, toPeg, middlePeg);
    System.out.println("Move disk from peg "+fromPeg+" to peg "+toPeg);
    TowersOfHanoi(n-1, middlePeg, fromPeg, toPeg);
}
```

4. *(25 points) Stooge Sort.* One fine day the three stooges were faced with the problem of sorting. After the customary back-slapping, name-calling, slipping and sliding, pie-in-the-face brain-storming session, Larry, Moe and Curly proposed the following "elegant" sorting algorithm!

```
STOOGE-SORT(A,i,j)
        if A[i] > A[j] then
                swap A[i] and A[j]
        if (i + 1) ≥ j then
                return
        k ← ⌊(j − i + 1)/3⌋ /* round down */
        STOOGE-SORT(A,i,j-k) /* sort first two-thirds */
        STOOGE-SORT(A,i+k,j) /* sort last two-thirds */
        STOOGE-SORT(A,i,j-k) /* sort first two-thirds again */
end
```

Assume that the algorithm does sort correctly. Now answer the following questions.

1. Write down the recurrence equation describing the worst-case running time for STOOGE-SORT. You can assume that n has the appropriate form so that you don't have to deal with floors and ceilings in the recurrence equation. (5 points)

2. Now use the divide-and-conquer recurrence master theorem to obtain an asymptotic upper bound on the running time. (10 points)

3. Which of the following sorting algorithms is STOOGE-SORT better than in the worst case? Insertion Sort, Quick Sort, Merge Sort. (10 points)

4. What should be your reaction towards the three stooges?

   (a) Give them a pat on the back.

(b) I am glad that you three are not in my study group.
(c) "Very funny," in a sarcastic tone of voice.

Creative

5. **(25 points) Heaps.**

   1. Does the following array represent a min-heap? (5 points)
      `A={1,15,8,25,16,10,12,30,31,17,8,11}`

   2. The operation HEAP-DELETE(A,i) deletes the item in node i from the max-heap A. Give an implementation of HEAP-DELETE that runs in $O(\lg n)$ worst-case time for an n-element max-heap. (15 points)

6. (a) (25 points) (10 points) Consider the following numbers (shown in binary). Show the operation of radixsort on these numbers when we consider these as base 4 numbers (instead of binary). That is, show the array after each invocation of counting sort.

```
0010

1000

1010

1100

0111

1001

0110

1011
```

(b) (15 points) Now consider the same numbers again. This time you also see their value in decimal alongside. The decimal values are given with assumption that the numbers are represented in *2's complement* notation. The leading bit is a 0 for positive numbers and a 1 for negative numbers. (*Refresher:* In 2's complement notation, positive numbers are represented with a leading 0 bit. To represent a negative number, take the reprensentation for the absolute value of the number. Then preform a bit-wise complement of the number and finally add 1.)

```
(+2) 0010
(-8) 1000
(-6) 1010
(-4) 1100
(+7) 0111
(-7) 1001
(+6) 0110
(-5) 1011
```

Does radixsort algorithm still sort these numbers correctly under the new interpretation of these numbers? If not, then how would you modify radixsort to fix the problem?