Name:_______________________________ ID:_______________________________

CSE 246 Analysis of Algorithms

Spring 2016 Midterm

Duration: 105 minutes

Q1	Q2	Q3	Q4	Q5	Q6	SUM
/16	/22	/20	/10	/20	/20	/100

Q-1. (2×8 = 16 pts) For each of the following, indicate whether it is true or false, by giving very short reasoning.

(i) $\log_2 n^2+1 \in O(n)$.

(ii) $\sqrt{n(n+1)} \in \Omega(n)$.

(iii) $n^{n-1} \in \Theta(n^n)$.

(iv) $\log_2 \sqrt{n}$ and $(\log_2 n)^2$ are of the same asymptotical order.

(v) 4^n has higher asymptotical order than $2^n + n^3$.

(vi) $3 \cdot \log_2 n^2$ has higher aymptotical order than $2 \cdot \log_3 \sqrt[3]{n}$.

(vii) For any function $f(n)$, it is true that $f(n) \in \Theta(f(2n))$

(viii) For any function $f(n)$, it is true that $f(n) \in \Theta(2f(n))$

Q-2. (a - 12 pts) How many lines, as a function of n does the following program print? Write a recurrence relation and solve it by backward substitution. You may assume n is a power of 2.

```
function f(n)

if n <= 1:
      print_line("Take it easy")
else:
      for i=1 to n
            f(n/2)
      end for
```

Name:_____________________________ ID:_____________________________

(b – 10 pts) What is the time complexity of the following function? Indicate your answer in $\Theta(\cdot)$ form.

```
void func2(int n) {
      int i = n;
      int x = 0;
      int count = 0;

      while (i > 1){
            x = x + 2;
            i = i/3;
      }
      for(int j = 1; j <= x; j++)
            for(int k = 1; k <= x; k++)
                  count = count + 1;
}
```

Q-3. (20 pts) Consider the following algorithm.

```
Algorithm Stooge-sort (A[0..n-1)
//Input: Array A of n numbers
//Output: A is sorted in increasing order
if n=2 and A[0]>A[1], then swap(A[0],A[1])
if n>2 then {
      Stooge-sort(A[0..ceil(2n/3)]) // sort first two-thirds.
      Stooge-sort(A[floor(n/3)..n]) // sort last two-thirds.
      Stooge-sort(A[0..ceil(2n/3)]) // sort first two-thirds again.
}
```

(a – 6 pts) Let T(n) denote the worst case number of comparisons (A[0]>A[1]) made for an input array of n numbers. Give a recurrence relation for T(n)

(b – 6 pts) Solve the recurrence relation using Master theorem.

(c – 8 pts) Is Stooge-sort correct? Prove your answer. If it is correct, would you use it in an application? Why?

Name:_____________________________ ID:_____________________________

Q-4. (10 pts) Consider the following decrease-and-conquer algorithm to check connectivity of a graph defined by its adjacency matrix.

```
Algorithm Connected(A[0..n − 1, 0..n − 1])
//Input: Adjacency matrix A[0..n − 1, 0..n − 1]) of an undirected graph G
//Output: 1 (true) if G is connected and 0 (false) if it is not

if n == 1 return 1 //one-vertex graph is connected by definition
else {
      if not Connected(A[0..n − 2, 0..n − 2]) return 0
      else {
            for j <- 0 to n − 2 do
                 if A[n − 1, j] == 1 return 1
            return 0
      }
}
```

This code does not work. Explain why! (This is a pseudocode, so do not say "syntax error"!)

Q-5. (20 pts) Given an unsorted array of n numbers, the problem is to determine the number of pairs whose product is equal to M. (For example, for the array 2,4,3,10,6,8,12 and M=24, the answer is "3" because 2*12, 3*8, and 4*6 are all equal to 24.) The answer is zero if there is no such pair.

(a – 7 pts) Design a brute-force algorithm for this problem. (Give a pseudocode) Describe its time complexity.

(b – 13 pts) Design a more efficient algorithm with O(nlogn) time complexity. (Give a step-by-step description. Don't give a pseudocode.)

Q-6 (20 pts) (a – 15pts) Given n points in the plane, design a Divide-and-Conquer algorithm to count the <u>number of pairs</u> of points such that the distance between the points is <u>at most twice</u> the distance between the closest pair of points. Explain the conquer step clearly. (Hint: First compute the closest pair, and you may assume that closest pair algorithm described in the class is given to you. You don't need to explain it again.)

(b – 5 pts) Write a recurrence relation for the time complexity of your algorithm and solve it using Master's Theorem.