Name:

COP 5407: Intro. to Algorithms Midterm Review; Spring 2017

---

1. **Short Questions**

   (a) [10] State and prove the correct relationship $(O, o, \Omega, \omega, \Theta)$ between the functions $f(n) = n^2 \log n + 2n$ and $g(n) = n^3 + 4n \log n$.

   (b) Solve the following recurrence relations using any of the 3 methods we have discussed in class:

   i. $T(n) = 2T(2n/3) + O(n)$
   ii. $T(n) = \frac{2}{3}T(2n) + O(n)$
   iii. $T(n) = 2T(2n/3) + O(n^2)$
   iv. $T(n) = 2T(n/2) + O(n^2)$
   v. $T(n) = 2T(n/4) + 1$
   vi. $T(n) = 2T(n/4) + \sqrt{n}$
   vii. $T(n) = 2T(n/4) + n$
   viii. $T(n) = 2T(n/4) + n^2$

   (c) The standard implementation of insertionSort (shown below) operates by inserting (in iteration j) $A[j]$ into its appropriate location in the sorted subarray $A[1 \ldots j-1]$. However, the right location is computed by a linear search (while-loop in lines 4 through 7), which has a worst-case time complexity linear in its length ($O(j)$). Can insertionSort be speeded up by replacing the linear search by a binary search with a logarithmic worst-case time complexity? What would be the time complexity of the modified insertionSort?

Algorithm 1 InsertionSort(A)

```
1: for j ← 2 to length[A] do
2:     key ← A[j]
3:     i ← j − 1                    ▷ Insert A[j] into sorted subarray A[1 ... j − 1]
4:     while i > 0 and A[i] > key do
5:         A[i + 1] ← A[i]
6:         i ← i − 1
7:     end while
8:     A[i + 1] ← key
9: end for
```

2.