Due Date: September 24, 2012

Total points: 46

1. (8 points)

   Answer the following by using the definitions for asymptotic notations:

   i) Prove that $10\sqrt{n} = O(n)$.

   ii) Is $2^n = \Theta(2^{n+1})$? Either prove or explain your answer.

2. (3 points)

   Express the function $f(n) = \frac{n^3}{100} - 10n^2 + n + 3$, in terms of Θ-notation. No need for any proof. Just state $\Theta(?)$.

3. (8 points)

   Order the following set of functions by their growth rates: N, $\sqrt{N}$, $N^{1.5}$, N^2, $N\ log\ N$, $N\ log\ log\ N$, $N\ log^2 N$, $N\ log(N^2)$, $2/N$, 2^N, $2^{N/2}$, 37, $N^2\ log\ N$, N^3. If there are functions that grow at the same rate, indicate them inside braces. There is no need to show proofs or give an explanation.

4. (6 points)

   State TRUE or FALSE for the following assertions. If FALSE give a counterexample as well.

   i) $f(n) = \omega(g(n))$ implies $g(n) = O(f(n))$

   ii) $\frac{O(n^2)}{O(n)} = \Theta(n)$

   (note: the terms on the left hand side are both big-Oh's)

5. (6 points)

   Consider the following algorithm to sort n integers in array $A[1 \ldots n]$. First, find the smallest element and swap it with $A[1]$. Next, find the second smallest element and swap it with $A[2]$. Repeat the process until all elements are sorted.

   Write a small pseudocode for this algorithm. Then state its worst-case and best-case running times using the Θ-notation.

6. (6 points)

A program takes 10 seconds for input size 1000 (i.e., n=1000). Ignoring the effect of constants, approximately how much time can the same program be *expected* to take if the input size is increased to 2000, under each of the following run-time complexities?

i) $O(N)$

ii) $O(N\ logN)$

iii) $O(N^2)$

In your answer show the logic of your derivation.

7. (9 points)

a) What is the run-time complexity of the following code? Your answer should be a tight bound for the worst-case – i.e., $\Theta(?)$). You can assume each primitive operation (arithmetic, print, assignment, etc.) takes $O(1)$ time (i.e., constant time).

---

0. Init(A[n]); // assume the array A is initalized in this step with n values, and that the cost of this initialization function as a whole is $\Theta(n)$.

```
1. FOR i = 1 TO n {
2.                 FOR j = 1 TO n {
3.                         print A[j];
4.                 }
5.         }
6. }
```

---

b) What is the memory complexity of the above code? Again, express your answer in $\Theta(?)$.

c) What is the run-time complexity of the following modified pseudocode?

0. Init(A[n]); // assume the array A is initalized in this step with n values, and that the cost of this initialization function as a whole is $\Theta(n)$.

```
1. FOR i = 1 TO n {
2.         IF (i == 1) THEN {
3.                 FOR j = 1 TO n {
4.                         print A[j];
5.                 }
6.         }
7. }
```