CMPS 256 — ADVANCED ALGORITHMS AND DATA STRUCTURES
Fall 2012 – 13
Quiz 1, Section 1
Monday November 12, 9:00 – 9:50 a.m.

Instructions.
This quiz is scored out of 100.
This quiz is open book and open notes: you can use the textbook, notes you have taken in class, and your homework solutions.
Show your work, as partial credit will be given.
You may use any algorithm that was covered in class. Just give the name of the algorithm and the chapter or page number in the book where the algorithm is given. If taken from your lecture notes, just say "lecture notes."

Please draw a horizontal line to separate each answer from the next.
Best of luck!

Problem 1 (25 points, 5 points each).

Give tilde approximations for the following. Give the simplest possible expression. Show your work.

1. $2^{3n} + (2^n)^4$
2. $(n^2 + n)/n^2$

State whether the following are true or false. Give a brief proof for your answer.

3. $n^3 \lg n = \Omega(n^{3.01})$
4. $\lg(n^n) = O(n)$
5. $(2n+3)(3n+4) \sim 3n^2$

Problem 2 (25 points).

Give the worst case running time of the following, expressed only in terms of n. You may express this as $\Theta(f(n))$, for suitable $f(n)$.

```
{$n > 0$}
$i := 1$;
while $i \leq n^2$
    $j := 1$;
    while $j \leq i$
        $j := j + 1$;
    endwhile
    $i := i + 1$;
endwhile
```

Problem 3 (50 points). You are given as input an array a that is *bitonic*: it consists of a non-increasing sequence of elements followed by a non-decreasing sequence of elements. Formally, you may assume that the following precondition holds initially, where $n = |a|$ is the size of a:

$$(\exists\, k : 0 < k < n-1 : \mathit{nonInc}(a[0,\ldots,k]) \wedge \mathit{nonDec}(a[k,\ldots,n-1])).$$

where $\mathit{nonInc}(a[\ell,\ldots,r] \triangleq (\forall\, i : \ell \le i < r : a[i] \ge a[i+1])$, and
$\mathit{nonDec}(a[\ell,\ldots,r] \triangleq (\forall\, i : \ell \le i < r : a[i] \le a[i+1])$.

You are to design an algorithm for sorting a. Your algorithm must have **worst case** running time in $O(n)$. Remember that worst case is a limit on the running time for **all** inputs, unlike average case running time.

Part a, (20 points). Give a design of your solution. This should be sufficiently detailed so that all algorithmic details are clear, and it would be straightforward to translate into executable code. You may use several levels of refinement if you wish.

Part b, (15 points). Give a proof that your algorithm correctly sorts a. The use of effect-comments (assertions) may help you.

Part c, (15 points). Prove that your algorithm has average case running time in $O(n)$.

You will receive **NO CREDIT** for an $O(n \lg n)$ or worse running time algorithm.

Sample Solutions.

Problem 1 (25 points, 5 points each).

Give tilde approximations for the following. Give the simplest possible expression. Show your work.

1. $2^{3n} + (2^n)^4$ **Sample solution.** $(2^n)^4$, since its the higher order term. So 2^{4n}.
2. $(n^2 + n)/n^2$ **Sample solution.** $(n^2 + n)/n^2 = 1 + 1/n$. The $1/n$ goes to 0, so its ~ 1.

State whether the following are true or false. Give a brief proof for your answer.

3. $n^3 \lg n = \Omega(n^{3.01})$ **Sample solution.** False, since $n^{0.01}$ grows faster than $\lg n$.
4. $\lg(n^n) = O(n)$ **Sample solution.** $\lg(n^n) = n \lg n$ which grows faster than $O(n)$, so false.
5. $(2n+3)(3n+4) \sim 3n^2$ **Sample solution.** $(2n+3)(3n+4)$ has high order term $6n^2$ which is not $\sim 3n^2$, so false.

Problem 2 (25 points).

Give the worst case running time of the following, expressed only in terms of n. You may express this as $\Theta(f(n))$, for suitable $f(n)$.

```
{$n > 0$}
$i := 1$;
while $i \leq n^2$
    $j := 1$;
    while $j \leq i$
        $j := j + 1$;
    endwhile
    $i := i + 1$;
endwhile
```

Sample solution. Running time given by the following summation, by putting the limits of the loop as the limits of the sum:

$$\sum_{i=1}^{n^2} \sum_{j=1}^{i} 1$$

the summand is 1 since inner loop body has constant running time. We could also have used a constant c.

The cost of $i := i + 1$ is ignored since its lower order.

The above simplifies to

$$\sum_{i=1}^{n^2} i$$

which further simplifies to $n^2(n^2 + 1)/2$, which is $\Theta(n^4)$.

Problem 3 (50 points). You are given as input an array a that is *bitonic*: it consists of a non-increasing sequence of elements followed by a non-decreasing sequence of elements. Formally, you may assume that the following precondition holds initially, where $n = |a|$ is the size of a:

$$(\exists\, k : 0 < k < n-1 : nonInc(a[0,\ldots,k]) \wedge nonDec(a[k,\ldots,n-1])).$$

where $nonInc(a[\ell,\ldots,r] \triangleq (\forall\, i : \ell \leq i < r : a[i] \geq a[i+1])$, and
$nonDec(a[\ell,\ldots,r] \triangleq (\forall\, i : \ell \leq i < r : a[i] \leq a[i+1])$.

You are to design and code an $O(n)$ method for sorting a. Your algorithm must have **worst case** running time in $O(n)$. Remember that worst case is a limit on the running time for **all** inputs, unlike average case running time.

Assume that levels 0 and 1 for the design are those that we presented in class for sorting.

Part a, (20 points). Give a design of your solution. This should be sufficiently detailed so that all algorithmic details are clear, and it would be straightforward to translate into executable code. You may use several levels of refinement if you wish.

Sample solution.
linear scan from $i = 0$ find first i s.t. $a[i] < a[i+1]$. let this be k.
reverse $a[0,\ldots,k]$
now have two nondecreasing array sections
use *merge* to merge into final sorted a

note that either section can be empty.

Part b, (15 points). Give a proof that your algorithm correctly sorts a. The use of effect-comments (assertions) may help you.

Sample solution. Omitted

Part c, (15 points). Prove that your algorithm has average case running time in $O(n)$.

Sample solution.
finding k is by linear search: $O(n)$ since body of loop compares $a[i]$ with $a[i+1]$.
reverse is $O(n)$ sing the obvious algorithm.
merge was shown to be $O(n)$ in class.