CMPS 256 — ADVANCED ALGORITHMS AND DATA STRUCTURES
Fall 2012 – 13
Quiz 1, Section 2
Monday November 12, 10:00 – 10:50 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. $n^4 + 3n^2(\lg n)^8$
2. $2^n + n^{100n}$
3. $(3n^3 \lg n + 6n^3)/n^2$

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

4. $n^4 = \Omega(n^{3.99} \lg n)$
5. $\lg(3^n) = O(2^n)$

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 $i < n$
    $j := 1$;
    while $j < i^2$
        $j := j + 1$;
    endwhile
    $i := i + 1$;
endwhile
```

Problem 3 (50 points).

You are given as input an integer array a all of whose elements have values 1, 2, or 3.

Formally, you may assume that the following precondition holds initially, where $n = |a|$ is the size of a:

$$(\forall\, i : 0 \leq i < n : a[i] = 1 \vee a[i] = 2 \vee a[i] = 3).$$

You are to modify selection sort to sort a with worst-case running time in $O(n)$. Your algorithm can change a using only swaps of elements of a, and can use only a constant amount of memory in addition to that for a itself.

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 worst-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. $n^4 + 3n^2(\lg n)^8$ **Sample solution.** n^4 since its the high order term
2. $2^n + n^{100n}$ **Sample solution.** n^{100n} since its the high order term
3. $(3n^3 \lg n + 6n^3)/n^2$ **Sample solution.** $3n \lg n$ since its the high order term

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

4. $n^4 = \Omega(n^{3.99} \lg n)$ **Sample solution.** true since $n^{0.01}$ grows faster than $\lg n$
5. $\lg(3^n) = O(2^n)$ **Sample solution.** true, since $\lg(3^n) = n \lg 3$ which is $O(2^n)$ since polynomials grow slower than exponentials.

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 $i < n$
    $j := 1$;
    while $j < i^2$
        $j := j + 1$;
    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:

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

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

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

which is $\Theta(n^3)$.

Problem 3 (50 points).

You are given as input an integer array a all of whose elements have values in $1, \dots, 3$.

Formally, you may assume that the following precondition holds initially, where $n = |a|$ is the size of a:

$$(\forall\, i : 0 \leq i < n : a[i] = 1 \vee a[i] = 2 \vee a[i] = 3).$$

You are to design and code a comparison-based algorithm for sorting a whose average case running time is in $O(n)$. Your algorithm can change a using only swaps of elements of a, and can use only a constant amount of memory in addition to that for a itself.

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.

```
sort(array a)                //4
{
//|a| > 0
split a into left section and right section

//invariant: left section is always ordered

while 1 elements remain in right section
  //invariant: split right section into left "scanned" part and right "unscanned" part
  //    no 1 elements in scanned part of right section, which is initially empty
  if last element of left section = 1 then extend left section
  else
      find next element=1 of right section by scanning from where last 1 element was found
      swap last element of left section with this 1 element of right section
      extend left section by one position

//no 1 elements are in right section and left section contains only 1 elements


while 2 elements remain in right section
  //invariant: split right section into left "scanned" part and right "unscanned" part
  //    no 2 elements in scanned part of right section, which is initially empty
  if last element of left section = 2 then extend left section
  else
      find next element=2 of right section by scanning from where last 2 element was found
      swap last element of left section with this 2 element of right section
      extend left section by one position

//no 1 or 2 elements are in right section and
// left section contains a block of 1 elments followed by a block of 2 elements

//a is ordered, i.e., a[i] <= a[i+1]
}
```

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

Sample solution. From the effect-comments given in the pseudocode for part (a), we have the following. After first while loop, left section consists of all the 1 elements in a and nothing else. After the second while loop, the left section consists of all the 1 elements in a followed by all the 2 elements in a. The right section therefore consists of all the 3 elements, since those are the only remaining elements. Hence a is sorted.

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

Sample solution.
Sum of number of iterations of outer while loops is at most n.
Inner loops scan to find the next element $= 1$ or $= 2$.
Each inner loop starts scan from where it left off last time, and so it scans at most n elements.
Hence each inner loop iterates at most n times *in total*, i.e., the sum of all iterations that the inner loop ever executes is n.
So total running time is $O(n)$.