CSE 100: Algorithm Design and Analysis
Midterm Exam 1

Spring 2019

Note: Please write down your name on *every* page. 8 pages in total including this cover. Time: 4:30-5:45pm Your score will be capped at 90 points. So, the maximum score you can earn is 90 points.

Problem	Points earned	Out of
1		20
2		4
3		6
4		5
5		5
6		5
7		10
8		20
9		10
10		15
Sum		Max 100
Final (after capping at 90)		Max 90

Name ______________________________

1. **[20 points]**. For each of the following claims, decide if it is true or false. *No* explanation is needed.

   (a) **[2 points]** $n \log n = O(n^2)$.
   True **False** **Sol.** True

   (b) **[2 points]** $\log \log n = O(\log n)$.
   True **False** **Sol.** True

   (c) **[2 points]** $\log^{50} n = O(n^{0.1})$.
   True **False** **Sol.** True

   (d) **[2 points]** $4^n = O(2^n)$.
   True **False** **Sol.** False

   (e) **[2 points]** $100^{100} = \Theta(1)$.
   True **False** **Sol.** True

   (f) **[2 points]** if $f = O(g)$, then $g = \Omega(f)$.
   True **False** **Sol.** True

   (g) **[2 points]** if $n^3 = \Omega(n^2)$.
   True **False** **Sol.** True

   (h) **[2 points]** $100 + 200n + 300n^2 = \Theta(n^2)$
   True **False** **Sol.** True

   (i) **[2 points]** $\lim_{n\to\infty} f(n)/g(n) = 5$, then $f(n) = O(g(n))$.
   True **False** **Sol.** True

   (j) **[2 points]** $\sum_{i=1}^{n} \Theta(i) = \Omega(n^2)$.
   True **False** **Sol.** True

2. **[4 points]** The following is a pseudocode of Insertion-sort. For instance $A[1 \dots 8] = \langle 7, 4, 2, 9, 4, 3, 1, 6 \rangle$, what is $A[1 \dots 8]$ just before the for loop starts for $j = 5$? **Sol.** $\langle 2, 4, 7, 9, 4, 3, 1, 6 \rangle$. If the answer is correct for $j = 4$ or $j = 6$, take off 0.5 pts.

```
INSERTION-SORT(A)
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted
           sequence A[1 .. j − 1].
    i = j − 1
    while i > 0 and A[i] > key
        A[i + 1] = A[i]
        i = i − 1
    A[i + 1] = key
```

3. (a) **[3 points]** Give an instance of size n for which the Insertion-sort terminates in $\Omega(n^2)$ time. **Sol.** e.g. any n numbers in decreasing order can be a solution.

   (b) **[3 points]** Give an instance of size n for which the Insertion-sort terminates in $O(n)$ time. **Sol.** e.g. any n numbers in increasing order can be a solution.

4. **[5 points]** Briefly explain the Random Access Model (RAM). **Sol.** Single processor (no parallel computing). Assumes each basic operation takes $O(1)$ time. Simple memory structure (and random memory access). Full points if a student explain two out of these three. If they get only one, they will earn 3 points.

5. **[5 points]** Let $T(n)$ denote the running time of Merge-sort on input of size n. In the following you can omit floor or ceiling.

```
MERGE-SORT(A, p, r)
1  if p < r
2      q = ⌊(p + r)/2⌋
3      MERGE-SORT(A, p, q)
4      MERGE-SORT(A, q + 1, r)
5      MERGE(A, p, q, r)
```

What is the running time of Line 3 (of Merge-Sort)?
What is the running time of Line 4 (of Merge-Sort)?
What is the running time of Line 5 (of Merge-Sort)?

Sol. $T(n/2)$, $T(n/2)$, $O(n)$ (or $\Theta(n)$). 1.5, 1.5, 2 points, respectively.

6. **[5 points]** Formally prove that $50n + 15 = O(n^2)$ using the definition of $O(\cdot)$. **Sol.** One possible answer: $50n + 15 \leq 65n^2$ for all $n \geq 1$.

7. **[10 points]** Prove the following: if $f = O(g)$ and $g = O(h)$, then it must be the case that $f = O(h)$.
   Sol. See Ch03 lecture.pdf (page 61).

8. **[20 points]** The following is a pseudocode of Insertion-sort. Prove its correctness via loop invariant. In other words, state the *loop invariant* and prove it using *Initialization, Maintenance, and Termination.*

```
INSERTION-SORT(A)
for j = 2 to A.length
    key = A[j]
    // Insert A[j] into the sorted
         sequence A[1 .. j − 1].
    i = j − 1
    while i > 0 and A[i] > key
         A[i + 1] = A[i]
         i = i − 1
    A[i + 1] = key
```

Sol. Loop invariant (8pts), inti. (1pt), Maintenance (10pts), Termination (1pt).

9. **[10 points]** The following is a pseudo-code of Selection-Sort. We would like to prove its correctness via loop invariant. State a *loop invariant*. Note that your loop invariant must be strong enough to lead to the correctness of the algorithm. You only need to state the loop invariant. (*No* need to show Initialization, Maintenance, or Termination.)

```
Selection-Sort(A)
n = A.length
for j = 1 to n-1
 smallest = j
  for i = j+1 to n
    if A[i] < A[smallest]
      smallest = i
  exchange A[j] with A[smallest]
```

Sol. Loop invariant: At the start of each iteration of the outer for loop, the subarray $A[1 \cdots j-1]$ consists of the $j-1$ smallest elements in $A[1 \cdots n]$, and is sorted.

10. **[15 points]** Prove that the following Merge-Sort is correct. That is, prove that Merge-Sort(A, p, r) sorts the subarray $A[p...r]$ in increasing order. You can assume that Merge(A, p, q, r) successfully merges two sorted arrays, $A[p...q]$ and $A[q + 1...r]$, into a sorted array $A[p...r]$. Please do not forget the *base* case.

```
MERGE-SORT(A, p, r)
if p < r
    q = ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    MERGE(A, p, q, r)
```

Sol. See the textbook or lecture slides. Base case: 3pts. Induction Step: 12pts.