Department of Computer Science & Engineering, Faculty of Engineering
CSC2100B/S: Data Structures **Written Assignment 1**
Spring 2011 Due: *18 Feb, 2011*

Name: ____________________
SID: ____________________

Section I: Revision Questions (10 points)
Put down your answers to the following simple revision questions.

1. Name the course lecturer: **Tang Wai Chung, Matthew** [ ] 1 pt

2. Visit the course newsgroup and find your answer: **RANDOM** [ ] 1 pt

3. The maximum subsequence sum for the following sequence is **10** . [ ] 1 pt

$$-7, 2, -1, 3, 4, -6, 5, 3$$

4. $O(N \lg N) + O(N^2) + O(N) =$ $O(N^2)$ [ ] 1 pt

5. Name the best algorithm for the *maximum subsequence sum* problem, in terms of running time complexity: **mss-online** . [ ] 1 pt

6. Before you can use binary search on a set of data, you have to first **sort** the data. [ ] 1 pt

7. **Internal** sorting occurs entirely inside the main memory of the computer. [ ] 1 pt

8. If the exchange of items are costly, we use try to use **selection** sort. [ ] 1 pt

9. **Bubble/Insertion/Merge** sort is an example of *stable* sorting algorithms. [ ] 1 pt

10. The worse case time complexity (in O-notation) of *quicksort* is $O(N^2)$. [ ] 1 pt

____

Section II: Short Questions (39 points)
Unless otherwise stated, you should answer the following questions in plain English or math expressions instead of code snippets.

1. Calculate the asymptotic expression for $T(N)$:

4 pts

$$T(N) = (N^2 + O(N) + O(1))(\log N^2 + O(N \log N) + O(1))$$

Solution:

$$\begin{aligned} T(N) &= (N^2 + O(N) + O(1))(\log N^2 + O(N \log N) + O(1)) \\ &= O(N^2 \log N) + O(N^3 \log N) + O(N^2) + O(N \log N) + O(N^2 \log N) + O(N) \\ &+ O(\log N) + O(N \log N) + O(1) \\ &= O(N^3 \log N) + O(N^2 \log N) + O(N^2) + O(N \log N) + O(N) + O(\log N) + O(1) \\ &= O(N^3 \log N) \end{aligned}$$

2. Show that

4 pts

$$O\big(f(N)g(N)\big) = f(N)O\big(g(N)\big)$$

Solution: Let $h(N) = O\big(f(N)g(N)\big)$, then for some $N \geq N_0$, we have

$$\begin{aligned} h(N) &\leq cf(N)g(N) \\ &\leq f(N) \cdot c \cdot g(N) \\ h(N)/f(N) &\leq c \cdot g(N) \\ h(N)/f(N) &= O(g(N)) \\ h(N) &= f(N)O(g(N)) \end{aligned}$$

assume that $f(N) \neq 0$.

3. Consider the following integer sequence S:

11 pts

$$3, 10, 5, 6, 13, 4, 9, 8, 2, 16$$

(a) (3 pts) Show your steps in using *selection sort* to sort S. (Lecture 03: p. 11)
Solution:

3	10	5	6	13	4	9	8	(2)	16
$\underline{2}$	10	5	6	13	4	9	8	(3)	16
$\underline{2}$	$\underline{3}$	5	6	13	(4)	9	8	10	16
$\underline{2}$	$\underline{3}$	$\underline{4}$	6	13	(5)	9	8	10	16
$\underline{2}$	$\underline{3}$	$\underline{4}$	$\underline{5}$	13	(6)	9	8	10	16
$\underline{2}$	$\underline{3}$	$\underline{4}$	$\underline{5}$	$\underline{6}$	13	9	(8)	10	16
$\underline{2}$	$\underline{3}$	$\underline{4}$	$\underline{5}$	$\underline{6}$	$\underline{8}$	(9)	13	10	16
$\underline{2}$	$\underline{3}$	$\underline{4}$	$\underline{5}$	$\underline{6}$	$\underline{8}$	$\underline{9}$	13	(10)	16
$\underline{2}$	$\underline{3}$	$\underline{4}$	$\underline{5}$	$\underline{6}$	$\underline{8}$	$\underline{9}$	$\underline{10}$	(13)	16
$\underline{2}$	$\underline{3}$	$\underline{4}$	$\underline{5}$	$\underline{6}$	$\underline{8}$	$\underline{9}$	$\underline{10}$	$\underline{13}$	16

(b) (3 pts) Show your steps in using *insertion sort* to sort S. (Lecture 03: p. 17)
Solution:

3	(10)	5	6	13	4	9	8	2	16
3	10	(5)	6	13	4	9	8	2	16
3	5	$\underline{10}$	(6)	13	4	9	8	2	16
3	5	6	$\underline{10}$	(13)	4	9	8	2	16
3	5	6	10	13	(4)	9	8	2	16
3	4	$\underline{5}$	$\underline{6}$	$\underline{10}$	$\underline{13}$	(9)	8	2	16
3	4	5	6	9	$\underline{10}$	$\underline{13}$	(8)	2	16
3	4	5	6	8	$\underline{9}$	$\underline{10}$	$\underline{13}$	(2)	16
2	$\underline{3}$	$\underline{4}$	$\underline{5}$	$\underline{6}$	$\underline{8}$	$\underline{9}$	$\underline{10}$	$\underline{13}$	(16)
2	3	4	5	6	8	9	10	13	16

(c) (3 pts) Show your steps in using *bubble sort* to sort S. (Lecture 03: p. 26)

Solution:

3	10	5	6	13	4	9	8	2	16
3	$\underline{5}$	$\underline{6}$	(10)	$\underline{4}$	$\underline{9}$	$\underline{8}$	$\underline{2}$	(13)	16
3	5	6	$\underline{4}$	$\underline{9}$	$\underline{8}$	$\underline{2}$	(10)	13	16
3	5	$\underline{4}$	(6)	$\underline{8}$	$\underline{2}$	(9)	10	13	16
3	$\underline{4}$	(5)	6	$\underline{2}$	(8)	9	10	13	16
3	4	5	$\underline{2}$	(6)	8	9	10	13	16
3	4	$\underline{2}$	(5)	6	8	9	10	13	16
3	$\underline{2}$	(4)	5	6	8	9	10	13	16
$\underline{2}$	(3)	4	5	6	8	9	10	13	16

(d) (2 pts) Which of the sorting algorithms above uses the most number of exchanges in sorting S? What is the number?

Solution: Bubble sort and insertion sort both use 19 exchanges.

4. Consider the following list of common playing cards which is originally sorted by suit:

5 pts

$$5\diamondsuit\ 8\diamondsuit\ K\clubsuit\ 8\clubsuit\ A\heartsuit\ 5\heartsuit\ K\heartsuit\ 10\heartsuit\ 10\spadesuit\ A\spadesuit\ 5\spadesuit$$

Use merge sort to sort the list in ascending order using rank as the key. (Lecture 03: p. 45) (Note: You should end with the list sorted in rank then suit.)

Solution:

$5\diamondsuit$	$8\diamondsuit$	$K\clubsuit$	$8\clubsuit$	$A\heartsuit$	$5\heartsuit$	$K\heartsuit$	$10\heartsuit$	$10\spadesuit$	$A\spadesuit$	$5\spadesuit$
$5\diamondsuit$	$8\diamondsuit$									
		$8\clubsuit$	$K\clubsuit$							
				$5\heartsuit$	$A\heartsuit$					
						$10\heartsuit$	$K\heartsuit$			
								$10\spadesuit$	$A\spadesuit$	
										$5\spadesuit$
$5\diamondsuit$	$8\diamondsuit$	$8\clubsuit$	$K\clubsuit$							
				$5\heartsuit$	$10\heartsuit$	$K\heartsuit$	$A\heartsuit$			
								$5\spadesuit$	$10\spadesuit$	$A\spadesuit$
$5\diamondsuit$	$5\heartsuit$	$8\diamondsuit$	$8\clubsuit$	$10\heartsuit$	$K\clubsuit$	$K\heartsuit$	$A\heartsuit$			
$5\diamondsuit$	$5\heartsuit$	$5\spadesuit$	$8\diamondsuit$	$8\clubsuit$	$10\heartsuit$	$10\spadesuit$	$K\clubsuit$	$K\heartsuit$	$A\heartsuit$	$A\spadesuit$

5. Illustrate quicksort with median-of-three pivot selection (underline your pivot) in sorting the following list of alphabets (Lecture 03: p.60):

5 pts

`Q U I C K S O R T`

Solution:

Q	U	I	C	K	S	O	R	T
K	O	I	C	$\underline{\text{Q}}$	S	U	R	T
C	I	$\underline{\text{K}}$	O					
C	I							
					S	R	$\underline{\text{T}}$	U
					R	S		
C	I	K	O	Q	R	S	T	U

6. Suppose that you are given a file of N records whose keys can only be 0 or 1. Describe an algorithm to sort the file and satisfies the following: (i) runs in $O(N)$ time, (ii) sorts in place, and (iii) use a constant amount of extra memory.

5 pts

Solution: The sorting algorithm for 0's and 1's resembles the partitioning procedure in *quicksort* (linear execution time):

- Keep two pointers i and j that points the beginning of the list and the end of the list respectively (constant memory).
- Then move i to right until $a[i] = 1$ and move j to the left until $a[j] = 0$.
- Exchange the elements (sort in place).
- Repeat until i and j cross each other.

7. (Bonus) Show that

$$N(\sqrt[N]{N} - 1) = O(\ln N)$$

5 pts

Solution: First consider that

$$\sqrt[N]{N} = e^{\ln N/N} = 1 + (\ln N/N) + O((\ln N/N)^2)$$

Thus

$$\begin{aligned} N(\sqrt[N]{N} - 1) &= N(\ln N/N + O((\ln N/N)^2)) \\ &= \ln N + O((\ln N)^2/N) \\ &= O(\ln N) \end{aligned}$$

as $(\ln N)^k/N$ tends to 0 as N tends to ∞ for constant k.

Section III: Long Question / Code Study (36 points)
Unless otherwise stated, you should answer the following questions in plain English instead of code snippets.

1. Study the following C function `f` that implements a useful algorithm **with a mistake**. 12 pts

```
1 int f(int a[], int n){
2     int i, m = 0;
3     for (i = 0; i < n; i++)
4         if (a[i] > m)
5             m = a[i];
6     return m;
7 }
```

(a) (2 pts) If the given array of integers is $\{5, 2, 4, 1, 6\}$, what value will be returned by the function?
Solution: 6

(b) (2 pts) Explain briefly what this algorithm is trying to do.
Solution: Find the maximum value among all elements in the array `a`.

(c) (3 pts) What is the mistake made in the implementation? Give a *perverse* input that reveals the mistake.
Solution: The coder assumes the maximum value is always greater than or equal to 0, and initialize `m` (the maximum value) to 0.
Any array containing all negative integers reveals the mistake.

(d) (3 pts) Put down a correct implementation in C (show code).
Solution: Modify line 2: initialize `m` to `a[0]`.
Modify line 3: begin looping with `i` = 1.

```
1 int f(int a[], int n){
2     int i, m = a[0];
3     for (i = 1; i < n; i++)
4         if (a[i] > m)
5             m = a[i];
6     return m;
7 }
```

(e) (2 pts) Use O-notation to express the running time $T(N)$ of this algorithm.
Solution: $T(N) = O(1) + O(N) \times O(1) + O(1) = O(N)$

2. The following C function implements another sorting algorithm that is not covered in our lecture, yet it is easy to understand. 14 pts

```
1  void gsort(int n, int a[]) {
2      int i = 1, tmp;
3      while (i < n){
4          if (i == 0 || a[i] >= a[i - 1])
5              ++i;
6          else {
7              tmp = a[i];
8              a[i] = a[i - 1];
9              a[i - 1] = tmp;
10             --i;
11         }
12     }
13 }
```

(a) (4 pts) Suppose the function is called with an array `a[]` $= \{5, 3, 2, 4, 1\}$, trace the code and show the contents of the array `a[]` whenever line 10 has finished.

Solution:

```
3 5 2 4 1
3 2 5 4 1
2 3 5 4 1
2 3 4 5 1
2 3 4 1 5
2 3 1 4 5
2 1 3 4 5
1 2 3 4 5
```

(b) (3 pts) What is the **best case** time complexity of this algorithm in sorting N items? Express and discuss your answer in O-notation.

Solution: The best case occurs when an array in the sorted order (e.g. $\{1, 2, 3, 4, 5\}$) is given. Then the algorithm compares $N - 1$ times through the array without any exchanges. Thus the best case time complexity $= O(N)$.

(c) (5 pts) What is the **worse case** time complexity of this algorithm in sorting N items? Express and discuss your answer in O-notation.

Solution: The worse case is an array with items in reversed order (e.g. $\{5, 4, 3, 2, 1\}$). Then the algorithm procedes as follows: exchanges the first and the second items, giving a sorted $[0, 1]$ subarray, then exchanges twice to give a sorted $[0, 2]$ subarray, and so on. In total there will be $1 + 2 + \ldots + (N - 1) = O(N^2)$ exchanges. Worse case time complexity $= O(N)$.

(d) (2 pts) Do you think this algorithm is better than the bubble sort? Give a reason.

Solution: Yes. The best case complexity is better than that of bubble sort.

3. Merge sort can be written without recursion. The following is the *bottom-up* version of merge sort based on the same merging routine `merge()` introduced in the lecture notes. 10 pts

```
1 void msort_bu(int *a, int *tmpa, int l, int r){
2     int i, m;
3     for (m = 1; m <= r - l; m = m + m)
4         for (i = l; i <= r - m; i += m + m)
5             merge(a, tmpa, i, i + m, MIN(i + m + m - 1, r));
6 }
```

You can assume `MIN(A, B)` is a function that finds the smaller number among A and B.

(a) (2 pts) If $l = 0$ and $r = 10$, how many times will the loop on line 3 be executed?

Solution: $m = 1, 2, 4, 8$. 4 times.

(b) (3 pts) List the calls to `merge()` when $l = 0, r = 10, m = 2$. (For example, `merge(a, tmpa, 0, 2, 3)`)

Solution:

```
merge(a, tmpa, 0, 2, 3)
merge(a, tmpa, 4, 6, 7)
merge(a, tmpa, 8, 10, 10)
```

(c) (5 pts) What is the time complexity of this algorithm in sorting N items? Present your detailed analysis in O-notation.

Solution: The for-loop on line 3 executes for $O(\lg N)$ times.
For each fixed m, the for-loop on line 4 examines each element on the array through `merge()`: $O(N)$ execution time (for line 4 + 5).
Hence the total time complexity is $O(N \lg N)$.