Name: Covers chapter 6 – 8, 11-13, 18	CS3112 Spring 2017 Midterm 2 California State University, Los Angeles Instructor: Jungsoo Lim

I pledge by honor that I will not discuss the contents of this exam with anyone.
Signed by _____ Date _____

Part I. (25 pts)

Suppose we have an array A = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

1. (5) How many comparisons are required to sort the array A if insertion sort algorithm is used?

---

2. (5) How many comparisons are required to sort the array A if selection sort algorithm is used?

---

3. (5) How many comparisons are required to sort the array A if quick sort algorithm is used?

---

4. (5) How many comparisons are required to sort the array A if merge sort algorithm is used?

```
LEFT(i)
        return 2i
RIGHT(i)
        return 2i + 1

MaxHeapify(A,  i)
        l = LEFT(i);
        r = RIGHT(i);

        if l <= A.heap-size and A[l] > A[i]
                largest = l;
        else
                largest = i;
        if l <= A.heap-size and A[r] > A[largest]
                largest = r;

        if largest != i
                exchange A[i] with A[largest]
                MaxHeapify(A, largest);

BuildMaxHeap(A)
        A.heap-size = A.length
        for i = floor(A.length/2) downto 1
                MaxHeapify(A, i)
```

1. (3) Run MaxHeapify(A, 3) on the array A={27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0}

2. (5) Which of the following array(s) is/are max heap?
   A = {72, 40, 70, 35, 30, 55, 65, 39}
   B = {59, 55, 40, 30, 50, 39, 35, 17}
   C = {62, 59, 42, 32, 39, 44, 13, 29}
   D = {78, 56, 20, 55, 57, 23, 15, 20}
   E = {74, 69, 32, 55, 40, 30, 29, 50}

3. (3) For an array of n records, what is the height of max heap tree?

4. (3) What is the effect of calling MaxHeapify(A,i) where i < A.heap-size/2 ?

5. (3) If we modify the for loop in BuildMaxHeap(A) to increase from 1 to floor(A.length/2) rather than decrease from floor(A.length/2) to 1, would the algorithm work?

6. (4) Rewrite MaxHeapify(A, i) that uses a loop instead of recursion.

7. (4) What is the running time of heapsort on an array A of length n that is already sorted in increasing order? What about decreasing order?

8. (4) What is the worst case running time of heapsort?

```
QuickSort(A, p, r)
        if p < r
        q = Partition(A, p, r)
        QuickSort(A, p, q - 1)
        QuickSort(A, q + 1, r)

Partition(A, p, r){
        x = A[r];
        i = p -1;
        for(j = p to r-1){
                if(A[j] <= x){
                        i++;
                        exchange A[i] with A[j]
                        }
                }
        Exchange A[i+1] with A[r];
        return i+1;
}
```

9. (3) Run Partition(A, 1, 10) on the array A = {6, 3, 10, 7, 4, 9, 2, 8, 1, 5}.

10. (3) What value of q does Partition return when all elements in the array A[p…r] has the same value? What would be a running time of QuickSort in this case?

11. (3) What is the best case running time of QuickSort?

12. (3) What is the worst case running time of QuickSort? How can we improve the worst case running time of QuickSort?

```
CountingSort(A, B, k){
        Let C[0…k] be a new array

        for i = 0 to k
                C[i] = 0

        for j = 1 to A.length
                C[A[j]] = C[A[j]] +1

        for i = 1 to k
```

```
        C[i] = C[i] + C[i-1];

for j = A.length downto 1
        B[C[A[j]]] = A[j]
        C[A[j]] = C[A[j]] - 1
```

13. (3) Run CountingSort(A,B, 6) on the array A = {6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2}. Show the contents of array C and B.

14. (3) What is the smallest possible depth of a leaf in a decision tree for a comparison sort?

Part II. (20 pts)

Please choose the best suitable algorithm from the list to solve each of the following problems.

A. Randomized algorithm
B. Insertion sort algorithm
C. Merge-Sort algorithm
D. Heap-Sort algorithm
E. Quick-Sort algorithm
F. Counting-Sort algorithm
G. Radix-Sort algorithm
H. Bucket-Sort algorithm
I. None of the Above

15. (5) Suppose we need to sort the population by age. What sorting algorithm would work best to sort the population?

16. (5) Suppose we need to sort the population by height. What sorting algorithm would work best to sort the population?

17. (5) Suppose we need to sort the ABC Bank customers by their account number. Assuming each customer has a unique account number, what sorting algorithm would work best to sort the account numbers?

18. (5) Suppose that we have an array with duplicate keys and the keys are associated with satellite data and the order of duplicate keys in the array must be kept while the array is sorted by the keys. What sorting algorithm would work best for this array?

Part III (20)

19. (10) Suppose that a dynamic set S is represented by a direct-address table T of length m. Describe a procedure that finds the maximum element of S. What is the worst-case performance of your procedure?

20. (10) When a direct-address table T is not feasible due to the size of Universe of keys, how can we solve this issue?

21. (10) What is a collision in hash table? How do we resolve the collisions in open addressing? How do we resolve it in chaining? What is the worst-case performance of hash table for standard operations: search/insert/delete?

22. (10) How does perfect hashing eliminate collision for static data?

23. (10) Does Universal hashing function eliminate collision for static data?

24. (10) Insert the keys 2, 3, 5, 7, 11, 13, 17, 19 to an empty binary tree. To search key 19, how many comparison do we need?

25. (10) What is the property of red-black tree? Is red-black tree a balanced binary search tree? What property of red-black tree enforce balancing the binary search tree? What is the maximum height of red-black tree for n nodes.

26. (10) Suppose we have branching factor 2 for B-tree and need to insert 10 keys, what is the maximum height and what is the minimum height of the tree?

Part IV. (35 pts)

Your algorithm can use any algorithm and data structure that we learned in class.

1. (10) You are given an array of n integers. Design an algorithm **Find-A-Pair** to find all unique pairs of elements (x, y) whose summation is S. Your algorithm must run in O(n log n) time.

2. (10) You are given multiple arrays of strings, where different string may have different number of characters. Design an algorithm **Make-A-Set** running in O(n log n) time, where the algorithm returns an union set of strings by combining all arrays and removing any duplicates.

3. (10) Suppose we have an array of n positive integers range from 0 to n – 1. Design an algorithm that sorts the array in O(n) time.