Algorithms Midterm Exam (Spring 2013)

總分：<u>115 %</u>

Name: ____________________________

Student ID #: ____________________

Question	**Score**
1 (30%)	
2 (15%)	
3 (22%)	
4 (10%)	
5 (10%)	
6 (28%)	
Total	

1. **[Simple questions: 30%]** Complete the following questions with simple answers.
   (a) [3%] Insertion sort is better than quicksort given an increasing sequence as the input, Yes or No? Why?

   (b) [3%] $O(n) + \Omega(n) = \Omega(n)$, Yes or No? Why?

   (c) [3%] $O(n^2) + \Omega(n) = \Omega(n)$, Yes or No? Why?

   (d) [5%] What is the minimum number of leaves in a decision tree model that is used for sorting n elements.

   (e) [5%] Define or use your own words to describe *simple uniform hashing*.

(f) [5%] Write down a complete shortest path for a decision tree model when we have four elements for comparison sorting.

(g) [6%] In a bank database, if we want to sort the records by name, then by gender, then by ID no., can you recommend an algorithm for this case? Some necessary detail should be given.

2. **[Recurrence equations: 15%]** Solve the recurrence equations in (a) and (b). You can assume the integer arguments for all cases. The answer should be an asymptotically tight solution. You are welcome to apply the master theorem when there is a need. After that, answer the question in (c)

   (a) $T(n) = 4T(n / 16) + \sqrt{n}$

   (b) $T(n) = 2T(n / 2 + 3) + n / 2 - 3$

(c) Give any example of function $f(n)$ for recurrence

$T(n) = 9T(n / 3) + f(n)$

where we can not use the master theorem to find the solution.

3. **[Heap: 22%]** We have the following code for heap manipulation. Answer the questions after that.

```
MAX-HEAPIFY(A, i)
1   l → LEFT(i)
2   r → RIGHT(i)
3   if  l ≤ heap-size[A] and A[l] > A[i]
          then  largest  ← l
          else  largest  ← i
6   if  r ≤ heap-size[A] and A[r] > A[largest]
          then  largest ← r
8   if  largest ≠ i
          then  SWAP(A[i], A[largest])
                MAX-HEAPIFY(A, largest)
```

```
BUIDE-MAX-HEAP(A)
1   heap-size[A] ← length[A]
2   for i ← ⌊length[A]/2⌋ downto 1
          do MAX-HEAPIFY(A, i)
```

(a) [6%] Is ⟨20, 18, 7, 4, 10, 5, 1, 2, 3, 8, 6⟩ a max-heap? Explain your answer if your answer is "yes", or make it a max-heap step-by-step by the above code if your answer is "no".

(b) [5%] Is a sequence of strictly decreasing order always a max-heap? Explain your answer as clear as you can.

(c) [5%] Given a sequence of strictly increasing order, can you suggest a fastest approach to build a max-heap.

(d) [6%] Compare two strategies to find the top-five elements in an un-sorted array: (1) Sort the array from the smallest to the largest one, then output the last five elements; (2) Build a max-heap, then extract the maximum for five consecutive times. Which one you prefer? Why?

4. **[Quicksort: 10%]** We have an alternative code for quicksort shown below. Different from the one discussed in class, we have QUICKSORT(*A, q, r*) rather than QUICKSORT(*A, q*+1, *r*) on line 4.

```
QUICKSORT($A, p, r$)
1       if  $p < r$
2           then $q \leftarrow$ PARTITION($A, p, r$)
3                QUICKSORT($A, p, q$-1)
4*               QUICKSORT($A, q, r$)
```

Can you comment this code from the correctness and efficiency point of view?

5. **[Sorting in Linear Time: 10%]** Answer the following questions. Your answer should be as clear as possible.
   (a) [5%] Show how to sort n binary integers in $O(n)$ time. That is, we have only 0 and 1 for sorting.
   (b) [5%] Show how to sort n integers in the range 0 to $n^4 - 1$ in $O(n)$ time.

6. **[Hashing : 28%]** Answer the following questions related to hashing.

   (a) [5%] Let us inserting n elements k_1, k_2, …, k_n to a hash table with collision resolved by linked list. Searching what element can be expected the most efficient one among all others? What is the time complexity for such search? Some brief explanation is necessary.

   (b) [5%] Following the previous question, the same question is asked again, but for open addressing.

   (c) [6%] Demonstrate what happens when we insert the keys 3, 19, 4, 14, 17, 1, 32, 16, 11, 10 into a hash table with collision resolved by linear probing. Let the table have 11 slots, and let the hash function be $h(k) = k \bmod 11$.

   (d) [7%] Now consider the hashing with collision resolved by double hashing for the same sequence and a hash table of size 11, what is your answer again? The hash functions are defined by $h_1(k) = k \bmod 11$ and $h_2(k) = 1 + (k \bmod 7)$.

   (e) [5%] If we modify the above hash function $h_2(k) = 1 + (k \bmod 7)$ and change it to $h_2(k) = 1 + (k \bmod 5)$? What will happen? The performance is better, worse, or remains to be the same? Why?