COMP 3711 Design and Analysis of Algorithms
Fall 2014 Midterm Exam

1. **Short-Answer Questions** (36 pts)

   1.1 (10 pts) We often need to use two parameters to measure the running time of an algorithm, such as k and n for non-comparison based sorting algorithms, and the relationship between k and n can be arbitrary. Consider the following functions involving n and k:

   $$(a)\ n;\ (b)\ n+k;\ (c)\ \max\{n,k\};\ (d)\ \min\{n,k\};\ (e)\ n\log k.$$

   In the following graph, draw an arrow "$\rightarrow$" from x to y if $x = O(y)$; draw a double arrow "$\leftrightarrow$" between x and y if $x = \Theta(y)$. You should not draw any arrow between x and y if their relationship can be arbitrary.

   $(a)\ n$

   $(b)\ n+k$ $\qquad$ $(e)\ n\log k$

   $(c)\ \max\{n,k\}$ $\qquad$ $(d)\ \min\{n,k\}$

   1.2 (6 pts) Put "$\forall$", "$\exists$", and "$\geq$" or "$\leq$" into the following blankets.

   The precise meaning of "any comparison-based sorting algorithm requires $\Omega(n\log n)$ comparisons" is:

   _____ correct comparison-based sorting algorithm, _____ $c > 0$, _____ $n_0 > 0$, such that _____ $n > n_0$, _____ an array of n elements for which the algorithm requires _____ $cn\log n$ comparisons to sort them.

   1.3 (20 pts) Solve the following recurrences. A correct answer gives you full credits; otherwise, showing the steps may gain you partial credits. Please give the answer using the $\Theta()$ notation. You may assume that n is a power of a for any constant $a > 1$ for your convenience. You may use the Master theorem (provided in the last page) whenever applicable.

   (a) $T(1) = 1$, and for all $n \geq 2, T(n) = 5T(n/3) + n$.
   (b) $T(1) = 1$, and for all $n \geq 2, T(n) = 5T(n/3) + n^2$.
   (c) $T(1) = 1$, and for all $n \geq 2, T(n) = 3T(n-1) + 1$.
   (d) $\star$ $T(n) = 1$ for $n \leq 2$, and for all $n \geq 2, T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n$.

2. **Guessing with and without Memory** (15 pts)

   Suppose we have a deck of n distinct cards (numbered from 1 to n), and we turn them over one at a time. Before turning over each card, you need to make a guess. What is the expected number of correct guesses, under each of the following two scenarios?

(a) Suppose you can't remember what has been turned over already, even the last one. So every time you simply guess a card from 1 to n uniformly at random.

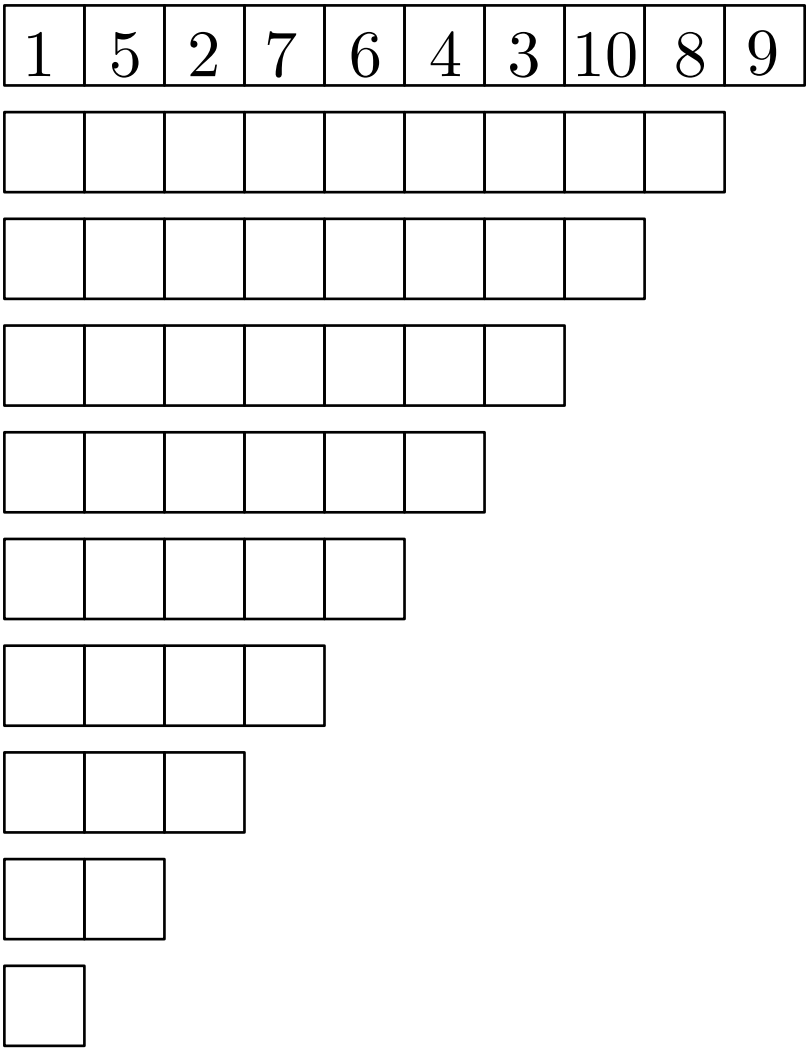
(b) Suppose you can remember ever card that has been turned over. Now every time you guess a card chosen uniformly at random from those not seen yet.

3. **Finding the Minimum on a Wave** (15 pts)

Suppose you are given an array A of size n. It starts with $A[1] = 0$, and increases to the maximum, and then decreases to the minimum, and increases again and finally gets back to $A[n] = 0$. The shape of the array is then like a single wave. For example, $A = [0, 2, 5, 8, 4, 3, 1, -3, -5, -2, 0]$ is such an array but $A = [0, 3, -2, 6, -3, 0]$ is not. Design an $O(\log n)$-time algorithm to find the minimum of the array. You may assume that all numbers in A are distinct.

4. **Heapsort** (14 pts)

Recall that in heapsort, we first insert all elements into a heap, and then do n Extract-Min operations. The following figure shows the contents of the heap (using the array implementation) after $n = 10$ elements have been inserted. Next, we will do n Extract-Min operations. Show the contents of the array after each Extract-Min (after the last Extract-Min, the array is empty and you don't have to draw this).



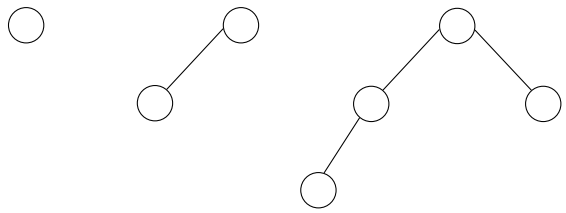
5. **Weight-Balanced Tree** (20 pts)

The AVL tree maintains its $O(\log n)$ height by balancing the heights of every two siblings. It is also possible to do so by balancing the weights. More precisely, the *weight* of a node u, denoted as $w(u)$, is the number of nodes in the subtree below u (including u). The weight of an empty tree is 0. We use u_L and u_R to denote the weight of u's left and right child, respectively. A node u is said to be *weight-balanced* if

$$\frac{1}{2} \leq \frac{w(u_L)+1}{w(u_R)+1} \leq 2.$$

A binary tree is weight-balanced if all of its nodes are weight-balanced.

(a) The following figures show the smallest weight-balanced trees of height $0, 1$, and 2, respectively. Please draw the smallest weight-balanced trees of height 3 and 4.



$n_0 = 1$ $n_1 = 2$ $n_2 = 4$

(b) Show that the height of a weight-balanced binary tree with n nodes is $O(\log n)$.