COMP 3711 Design and Analysis of Algorithms
Spring 2015 Midterm Exam

1. **Short-Answer Questions** (35 pts)

   1.1 (10 pts) Arrange the following functions in asymptotic ascending order (e.g., n, n^2, n^3)

   $$(a)\ n;\ (b)\ n^2/\log^3 n;\ (c)\ \sqrt{n}\cdot\log^4 n;\ (d)\ 10^{100};\ (e)\ \log n.$$

   1.2 (3 pts) When the input array is already sorted, insertion sort takes $O(n)$ time, which breaks the $\Omega(n \log n)$ sorting lower bound. Is this a contradiction? Why or why not?

   1.3 (6 pts) What is the fastest known algorithm for sorting an array of n elements for each of these cases (give both the algorithm and the corresponding running time):

   (a) when the elements are all 0's and 1's;
   (b) when the elements are integers in the range from 0 to n^3;
   (c) when the elements are real numbers.

   1.4 (16 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 $n \geq 2, T(n) = T(n-1) + 2n$.
   (b) $T(1) = 1$, and for all $n \geq 2, T(n) = 4T(n/3) + n$.
   (c) $T(1) = 1$, and for all $n \geq 2, T(n) = 2T(n/2) + \sqrt{n}$.
   (d) $T(1) = 1$, and for all $n \geq 2, T(n) = T(n-1) + \frac{1}{n}$.

2. **Rotated sorted array** (20 pts)

   Suppose you are given a sorted array A of n distinct numbers that has been rotated k steps, for some *unknown* integer k between 1 and $n-1$. That is, $A[1..k]$ is sorted in increasing order, and $A[k+1..n]$ is also sorted in increasing order, and $A[n] < A[1]$. The following array A is an example of $n = 16$ elements with $k = 10$.

   $A = [9, 13, 16, 18, 19, 23, 28, 31, 37, 42, 0, 1, 2, 5, 7, 8]$.

   (a) (15 pts) Design an $O(\log n)$-time algorithm to find the value of k.
   (b) (5 pts) Design an $O(\log n)$-time algorithm that for any given x, finds x in the array, or reports that it does not exist. [You may do part (b) assuming you have solved part (a), even if you can't.]

3. **Max-Heap** (10 pts)

   Given an array as follows, show the content of the array after we have run `Build-Max-Heap` on it.

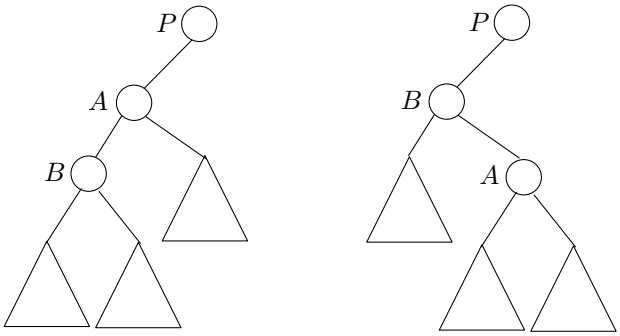
8	4	2	3	6	5	7	10	1	9

4. **Sorting strings** (10 pts)

   Recall that in Written Assignment 2, you gave an $O(n)$-time algorithm for sorting a collection of variable-length strings with a total of n characters. One student offered the following solution: First pad '\0' (the character with ASCII code 0) to the right of each string so that all strings have equal length, and then run radix sort on all strings. What's the worst-case running time of this algorithm? Give an input on which this running time is attained.

5. **Binary search tree** (10 pts)

   The following figure shows the rotation of the AVL-tree for the "left-left" case. Suppose each node in the AVL-tree has a `left` pointer, a `right` pointer, as well as an extra `size` field that stores its subtree size (so as to support rank queries). Recall that the subtree size of a node x is the number of nodes below x, including x itself. Given the pointers to nodes A, B, and P (assuming A is the left child of P), write pseudocode to implement this rotation, i.e., how the `left`, `right`, `size` fields are updated for all the affected nodes.



6. **Lazy hashing** (15 pts)

   Suppose you want to implement a hash table but are too lazy to implement either chaining or open addressing. So you decide to just use the following simple strategy: To insert an element x, you store it at position $A[h(x)]$; if $A[h(x)]$ is already occupied, you just throw x away. Suppose you insert a total of n elements into a hash table of size n. Do the following analyses of this lazy strategy under the uniform hashing assumption (derive the exact expressions in terms of n, not just asymptotic results):

   (a) What's the probability that the i-th inserted element is thrown away? [Hint: First compute the probability that it is *not* thrown away.]

   (b) What's the expected number of elements thrown away in total?