COMP 3711 Design and Analysis of Algorithms
Fall 2012 Final Exam

1. **Quick-Answer Questions** (15 pts)

   1.1 (6 pts) Please arrange the following functions in asymptotic ascending order (e.g., n, n^2, n^3): (a) n; (b) $\log n$; (c) $\log \log n$; (d) $\log^* n$; (e) $2^{2 \log n}$.

   1.2 (5 pts) Given n integers in the range 0 to n^3, what is the fastest sorting algorithm (asymptotically) to sort them?

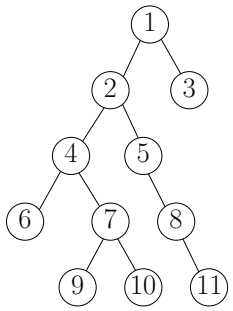
   1.3 (4 pts) Put "$\forall$" and "$\exists$" into the following blankets.

   The precise meaning of "the problem can be solved in $O(n)$ time" is:

   _____ a correct algorithm for the problem, _____ $c > 0$, _____ $n > 0$, such that _____ input of size n, the running time of the algorithm on the input is at most cn.

2. **Longest Path in a Tree** (10 pts)

   Design an algorithm that, given a binary tree, finds the longest path in the tree. For example, in the tree below, the longest path is 9-7-4-2-5-8-11. Your algorithm just needs to output the two endpoints of the path, i.e., 9 and 11 in this example. For full credits, your algorithm should run in $O(n)$ time, where n is the number of nodes in the tree.



3. **Shortest Path on an Chessboard** (10 pts)

   In the game of chess, the Knight can move two squares horizontally and one square vertically, or two squares vertically and one square horizontally. The complete move therefore looks like the letter "L". You are given two squares on an $n \times n$ chessboard. Design an algorithm to determine the shortest sequence of knight moves from one square to the other. For full credits, your algorithm should run in $O(n^2)$ time.

4. **Bottleneck Spanning Tree** (10 pts)

   A *bottleneck spanning tree* T of an undirected, weighted graph G is a spanning tree of G whose largest edge weight is minimum over all spanning trees of G. We say that the value of the bottleneck spanning tree is the weight of the maximum-weight edge in T. Give a linear-time algorithm that, given a graph G and an integer b, determines whether the value of the bottleneck spanning tree is at most b.

5. **Non-adjacent Elements in an Array** (15 pts)

   Given an array of n positive numbers $A[1], A[2], \ldots, A[n]$. Our goal is to pick out a subset of non-adjacent elements whose sum is maximized. For example, if the array is $(1, 8, 6, 3, 6)$, then the elements chosen should be $A[2]$ and $A[5]$, whose sum is 14.

(a) (5 pts) Give an example to show that the following algorithm does not always find a subset of non-adjacent elements whose sum is maximized.

```
Start with S equal to the empty set
While some elements remain in A
  Pick the largest A[i]
  Add A[i] to S
  Mark A[i] and its neighbors as deleted
Endwhile
Return S
```

(b) (10 pts) Give an algorithm based on dynamic programming that finds a subset of non-adjacent elements in A whose sum is maximized. For full credits, the running time should be $O(n)$. All elements in A are positive.

6. **Longest Balanced Subsequence** (15 pts)

A string of parentheses is said to be balanced if the left- and right-parentheses in the string can be paired off properly. For example, the strings (()) and ()() are both balanced, while the string ((()))(is not. Given a string S of length n consisting of parentheses, design an algorithm to find the longest subsequence of S that is balanced. For this problem, you will gain full marks as long as your algorithm is correct and runs in polynomial time.

7. **Greedy Algorithm** (15 + 10 pts)

A teacher assigns a homework with n questions at the beginning of the first day of class. Each homework question carries a certain number of marks, if submitted on or before a specified deadline; otherwise it earns no marks. Each homework question takes exactly one day to complete. Your task is to find a schedule to finish the homework questions so as to maximize marks earned.

For instance, suppose there are seven questions with deadlines and marks as follows:

Problem	1	2	3	4	5	6	7
Deadline	1	1	3	3	2	2	6
Marks	6	7	2	1	4	5	1

Then the maximum number of marks one can obtain is 15, which can be achieved by completing the problems in the order of 2, 6, 3, 1, 7, 5, 4. Note that there are also other schedules that obtain the same marks.

For this problem, you will gain full marks as long as your algorithm is correct and runs in polynomial time. You get 10 bonus marks if your algorithm runs in time $O(n \log n)$.

8. **Huffman Coding** (10 pts)

What is the optimal Huffman code for the following set of frequencies, based on the first 8 Fibonacci numbers?

a	b	c	d	e	f	g	h
1	1	2	3	5	8	13	21

Can you generalize your answer to find the optimal code when the frequencies are the first n Fibonacci numbers? Recall that the Fibonacci numbers are defined inductively as $f_1 = f_2 = 1, f_n = f_{n-2} + f_{n-1}$ for $n \geq 3$. You need to justify your answer.