B1. Algorithms: Search Trees(25 points)

1. (6 pts) Define the term "binary search tree."
2. (9 pts) Describe two tree representations (not necessarily binary) for inserting into a search tree so that the tree is maintained in an "approximately balanced" state. "Approximately balanced" means that any sequence of $n > 0$ insertions and/or searches can be done in $\mathcal{O}(n \lg n)$ time. Describe the relative merits of the two representations.
3. (10 pts) Give pseudo-code for the insertion algorithm for one of the representations described in the previous part. Be sure to clearly state which representation is being used and clearly define all variables etc. used in the code.

B2. Algoritms: Sorting(25 points)

For each of parts (1), (2), and (3) below,

(a) Give the algorithm in pseudo-code.

(b) Give good scientific reasons for your algorithm choice.

(c) Give a tight asymptotic running time upper bound for the algorithm.

1. Your employer asks you to recommend a fast general purpose algorithm for sorting arrays of randomly distributed (that is, each possible permutation of the input integers is equally likely) integers.
2. Your employer asks you to recommend a fast algorithm for sorting arrays of integers that are partially sorted.
3. Your employer asks you to recommend a fast algorithm for sorting "small" arrays of integers.

B3. Algorithms: Addition (25 points)

The *subset-sum problem* is, for a set of integers $S = \{s_1, s_2, \ldots, s_n\}$, and integer t, to determine if a subset of S adds up to t.

1. (5 pts) Define the class of problems NP, and show that subset-sum is in NP.
2. (8 pts) Explain what it means for a problem to be NP-complete and show that subset-sum is NP-complete. You may, for instance, take advantage of the fact that the *partition problem* is NP-complete. The partition problem is to determine if there is a subset S' of a given set S of integers such that the sum of S' is exactly half of the sum of S.
3. (8 pts) Give a deterministic algorithm for the subset-sum problem which runs in worst case time $\mathcal{O}(n^c t)$, for some constant c.
4. (4 pts) Explain why a solution to part (3) above, together with parts (1) and (2), is not a proof that P = NP.

B4. Algorithms: Multiplication (25 points)

1. (15 pts)
   Explain thoroughly an algorithm which has arithmetic complexity better than $\mathcal{O}(n^2)$ for the multiplication of two polynomials of degree less than n. Give and explain the worst case computing time as well as demonstrating the correctness of your algorithm. For simplicity you can restrict attention to the case that n is a power of 2.
   For instance, you could exploit these facts.
   Let $n = 2k$.
   (1) A polynomial $A(x) = \sum_{i=0}^{2k-1} a_i x^i$ can be written in terms of it's even part A_e and odd part A_o,

$$A_e(x) = \sum_{i=0}^{k-1} a_{2i} x^i, \quad A_o(x) = \sum_{i=0}^{k-1} a_{2i+1} x^i,$$

   as $A(x) = A_e(x^2) + xA_o(x^2)$.
   (2) If $A(x)$ and $B(x) = \sum_{i=0}^{2k-1} b_i x^i$ are polynomials, then

$$A_e(x)B_o(x) + A_o(x)B_e(x) =$$

$$[A_e(x) + A_o(x)]\,[B_e(x) + B_o(x)] - A_e(x)B_e(x) - A_o(x)B_o(x).$$

2. (10 pts) Strassen showed that 2×2 matrices can be multiplied using 7 element multiplications and 18 element additions. Explain how this can be used to create a multiplication algorithm for $n \times n$ matrices that has arithmetic complexity $\mathcal{O}(n^{2.81})$.