CSE 326
Autumn 2005
Assignment 1
Due 10/5/05

For all algorithm and data structure design problems please provide elegant pseudocode and an adequate explanation of your methods. If is often helpful to include small examples demonstrating the method. Put your name at the top of each sheet of paper that you turn in.

1. 1.9 a, page 13.

2. The classic way to evaluate a polynomial is called Horner's rule which can be stated recursively as follows. Let $p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$. To compute $p(c)$ for some constant c, first evaluate $q(c)$ where $q(x) = a_1 + a_2x + \cdots + a_nx^{n-1}$ recursively, then $p(c) = a_0 + cq(c)$.

   (a) Provide a base case for this method, and show the method works mathematically.

   (b) For a polynomial of degree n, as a function of n, how many additions and how many multiplications are used to evaluate the polynomial in Horner's rule.

   (c) Provide an elegant pseudocode function for Horner's rule where the data coefficients are stored in an array A[0..n], with A[i] containing a_i.

3. Consider the following general recurrence $T(1) \leq d$ and $T(n) \leq aT(n/b) + cn$ for $n \geq 1$. Show that:

   (a) If $a < b$ then $T(n) = O(n)$.

   (b) If $a = b$ then $T(n) = O(n \log n)$.

   (c) If $a > b$ then $T(n) = O(n^{\log_b a})$ (n to the power log base b of a).

   You may assume that n is a power of b in your argument. A very good argument would determine the constants and low order terms hidden by the big O notation.

4. Generally speaking, computers do not have built-in the exponentiation operator, m^n, for integers m and n. One way to compute exponentiation is iteratively multiply m into a running product n times. This would have $n - 1$ multiplications. Design a recursive algorithm that computes m^n using only $O(\log n)$ multiplications. For simplicity, assume that integer multiplication has infinite precision. Don't forget to explain why your algorithm is correct and has $O(\log n)$ multiplications.