Data Structures and Algorithms

Assignment #1

Due: Friday January 17

1. (10 points)
(a) (5 points). Write a pseudocode function for the problem in Weiss 1.5.
(b) (5 points). Prove by induction that your algorithm is correct.

2. (7 points) Weiss 2.1

3. (14 points)
(a) (3 points) Weiss 2.14 part (a)
(b) (3 points) Weiss 2.14 part (c)
(c) (5 points) Write in pseudocode a recursive function to compute a polynomial using Horner's rule. The arguments to the function are (i) a node pointer which points to a linked list of records with each record having two fields: an integer coefficient and a pointer to the next record. The head of the list will be a node with the coefficient a0 (corresponding to a[0] in Weiss 2.14), its successor will be a node with the coefficient a1 (corresponding to a[1]) etc. and (ii) an integer value for x. The function should return an integer value.
(d) (3 points) Would you advocate Horner's rule (either iterative or recursive) if your polynomial was dense, i.e., there would be few coefficients with the value 0.
Same question for the case where the polynomial is sparse, i.e., few coefficients are non-zero.

4. (15 points)
In order to check if a number is divisible by 3, we can add the individual digits and if the sum is divisible by 3, then the original number is divisible by 3.

For example, consider the number 6534. The sum of the individual digits is 18 (6+5+3+4) which is divisible by 3 (since 1 + 8 = 9 which is divisible by 3), so 6534 is also divisible by 3.

Write a pseudocode algorithm (function) to check whether a long positive integer represented in a linked list fashion as described in the lectures is divisible by 3 (cf. Lecture 3, slides 20 and following). Since the sign is always assumed to be positive, there is no need for a "sign" element; however you should be able to represent the value "0". Your *divisible_by_3* function should have one argument which is a node pointer to a linked list of digits (the least significant digit is the head of the list) and return a boolean value: true if the integer is divisible by 3 and false otherwise.

The constraints on the algorithm are as follows:
(i) Only the digits 0, 3, 6, and 9 are known to be divisible by 3.
(ii) The original long integer should not be destroyed.
(iii) The "intermediate sums" like 18 in the above example must be generated and must be in the form of long integers (this smells like the use of recursion should be handy; and yes we know that we could accumulate the sum directly but this is not the point of this exercise. Please follow the rules).