Assignment 2

Q1. Suppose you have an unsorted array A[1..n] of elements, and this array cannot be sorted. For example this can be array of JPEG images that you can only compare for equality/inequality. An element of this array is called dominant if it appears in the array more than half the time. For example in [a, b, a, a] the dominant element is a, but arrays [a, b, a, c] and [a, a, b, c] have no dominant element. You want to find a dominant element in array A. Straightforward approach of checking if A[i] is a dominant for all i = 1, . . . , n will have run-time in $\Theta(n^2)$ which is too slow. Consider the following approach to finding a dominant element: recursively find the dominant element y in the first half of the array and the dominant element z in the second half of the array, and combine results of recursive calls into the answer to the problem.

(a) show that if x is a dominant element in A then it has to be a dominant element in the first half of the array or the second half of the array (or both). [5 points]

(b) Using observation of part (a) give a divide-and-conquer algorithm to find a dominant element, that runs in time O(n log n). Detailed pseudocode is required. Be sure to argue correctness and analyze the run time. If given array has no dominant element, return FAIL. [5 points]

Q 2. An array A of n distinct integers A1, A2, . . . , An is known to have the following property: elements follow in descending order up to a certain index p where 1 < p < n and then follow in ascending order:

$A_1 > A_2 > \ldots > A_{p-1} > A_p < A_{p+1} < \ldots < A_n$.

Give an efficient algorithm (analogue of non-recursive binary search) to find the index of smallest value in this array (i.e., to find p). Note: the worst case running time of your algorithm must be in o(n), so simple scanning from left to right is not going to work. Detailed pseudocode is required. [5 points]

Q 3. A singly linked list contains n − 1 strings that are binary representations of numbers from the set {0, 1, . . . n − 1} where n is an exact power of 2. However, the string corresponding to one of the numbers is missing. For example, if n = 4, the list will contain any three strings from 00, 01, 10 and 11. Note that the strings in the list may not appear in any specific order. Also note that the length of each string is lgn, hence the time to compare two strings in O(lgn). Write an algorithm that generates the missing string in O(n). [5 points]