CMPS 101
Winter 2009
Homework Assignment 1

1. (1 Point) p.27: 2.2-2
Consider sorting n numbers stored in array A by first finding the smallest element of A and exchanging it with the element in $A[1]$. Then find the second smallest element of A and exchange it with $A[2]$. Continue in this manner for the first $n-1$ elements of A. Write pseudo-code for this algorithm, which is known as *selection sort*. What loop invariant does this algorithm maintain? Why does it need to run for only the first $n-1$ elements, rather than for all n elements? Give the best-case and worst-case running times of selection sort in Θ-notation.

2. (1 Point) p.37: 2.3-5
Referring back to the searching problem (see Exercise 2.1-3), observe that if the sequence A is sorted, we can check the midpoint of the sequence against v and eliminate half of the sequence from further consideration. *Binary search* is an algorithm that repeats this procedure, halving the size of the remaining portion of the sequence each time. Write pseudo-code, either iterative or recursive, for binary search. Argue that the worst-case running time of binary search is $\Theta(\lg n)$.

3. (4 Points) p.39: 2-4abcd
Let $A[1 \cdots n]$ be an array of n distinct numbers. If $i < j$ and $A[i] > A[j]$, then the pair (i, j) is called an *inversion* of A.

a. (1 Point) List the five inversions of the array $\langle 2, 3, 8, 6, 1 \rangle$.

b. (1 Point) What array with elements from the set {1, 2, 3, ….., n} has the most inversions? How many inversions does it have?

c. (1 Point) What is the relationship between the running time of insertion sort and the number of inversions in the input array? Justify your answer.

d. (1 Point) Give an algorithm that determines the number of inversions in any permutation of n elements in $\Theta(n \lg n)$ worst-case time. (Hint: Modify merge sort.)