Design & Analysis of Algorithms - Spring 2012
Mid Term 1

March 03, 2012 **Time: 90 min**

Q1. (15)
Devise an O(n) algorithm that determines the intersection of two sets of integers. The numbers in the sets may be positive and/or negative. The intersection of two sets is a set, possibly empty, that contains the common elements of the two sets.
First explain your algorithm in English and then write C++ code. Also derive the time complexity of code written in C++.

Q2. (5+5)
Find **maximum** and **minimum** items in a sorted list of n elements using **Divide and Conquer** strategy. First explain the algorithm in words and then write a recursive function to implement the algorithm. Also develop the recursive equation for the function and solve it.
(Note: The function should find out both the maximum and minimum function simultaneously. You should not write two functions --- one for minimum and the other for Maximum.)

Q3. (5+5)

(For sections A,B &C)

```
BUBBLESORT (A)
1 for i = 1 to A.length - 1
2 for j = A.length downto i-1
3 if A[ j ] < A[ j - 1]
4       exchange( A[ j ],  A[ j – 1] )
```

(For section D)

```
SelectionSort(A)
1. n = length[A]
2. for j = 1 to n − 1
3.      smallest = j
4.      for i = j + 1 to n
5.              if A[i ] < A[smallest]
6.                      smallest = i
7.      exchange (A[ j ], A[smallest])
```

a. State precisely a loop invariant for the **for** loop in lines 2–4 (lines 4-6), and prove that this loop invariant holds. Your proof should use the structure of the loop invariant proof presented in this chapter.

b. Using the termination condition of the loop invariant proved in part (a), state a loop invariant for the **for** loop in lines 1–4 (line 1-7) that will allow you to prove that BubbleSort(SelectionSort) sorts the array correctly. Your proof should use the structure of the loop invariant proof presented in this chapter.