Design & Analysis of Algorithms - Spring 2013
Mid Term 1

February 26, 2013 **Time: 90 min**

Q1. (15)
Suppose you have an unsorted array A of colors *red*, *white* and *blue*. You want to sort this array so that all *reds* are before all *whites*, followed by all *blues*. Only operations available to you for this purpose are: equality comparison A[i] == c where c is one of the three colors, and swap(i, j) which swaps the colors at indices i and j in A. Write an algorithm to sort this array in O(n).
First explain your algorithm in plain English and then code it.
(**Note: You cannot use an extra array in the solution.**)

Q2. (15)
You are given a very large array (you can assume it's of indefinite size); the first n entries of the array contain distinct integers in sorted order, after that all entries contain ∞. You DO NOT know the value of n. Devise an $O(lgn)$ time algorithm to search for an element key in this array.
(**Note**: The input to your program consists of a pointer to the beginning of the array, and the integer key.)

Q3. (5+5)

```
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 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 2-7) that will allow you to prove that SelectionSort sorts the array correctly. You should use the structure of the loop invariant proof presented in this chapter.