Design and Analysis of Algorithms

Sessional 1, Fall 2013

Date: September 23, 2013 **Marks: 45** **Time: 90 mins.**

Q1. [5+10] An array of n elements contains all but one of the integers from 1 to n+1.

i) Give the best algorithm you can for determining which number is missing if the array is sorted, and analyze its worst case asymptotic running time.

ii) Give the best algorithm you can for determining which number is missing if the array is not sorted, and analyze its worst case asymptotic running time.

Q2. [5 + 5] Analyze the running time of Quick Sort algorithm in the case where the pivot element always divides the array (at each step) into two potions of sizes 70% and 30% of the original.

i) Write a recurrence to describe the running time in this situation.

ii) Solve the recurrence to find an upper bound (Big O).

Q3. [5+5] Perform a step count analysis on the following and give tight bounds on each. Assume n is an exact power of 2.

i)
```
while(n >= 1){
        for(int j = n; j >= 1 ; j--){
                //do something in O(1)
        }
                n = n * 1/2;
}
```

ii)
```
for(int i=1; i <=n; i++){
        for(int j = i ; j >= 1 ; j = j/2){
                //do something in O(1)
        }
}
```

Q4. [5+5] A stable sort is one in which elements with same key values retain their relative order in the original array after sorting. For example, if elements x, and y, both have key equal to 5, and x appears before y in the array, then after sorting x still appears before y.

i) Is insertion sort (as implemented in class) a stable sort? (Don't write any code here, simply give a yes/no answer and explain why).

ii) How can we make sure that Merge Sort behaves like a stable sort? Where exactly in the code do we ensure this? (**Note:** You don't have to reproduce the code for Merge Sort, simply mention the relevant lines and how they ensure stability.)