Birla Institute of Technology & Science - Pilani, Hyderabad Campus
Second Semester 2014-2015
CS F211 / IS F211 : Data Structures and Algorithms
Test 1

Type: Closed **Time: 60 mins** **Max Marks: 60** **Date: 20.02.2015**

All parts of the same question should be answered together.

1.a. Is $2^n \in \Theta(3^n)$? Explain why or why not. [2 Marks]

1.b. Find a growth rate that squares the run time when we double the input size. That is, if T(n) = X, then $T(2n) = X^2$ (or equivalently if T(n) = O(g(n)) then prove that $T(2n) = O(g(n)^2)$ [3 Marks]

1.c. Find a growth rate that cubes the run time when we double the input size. That is, if T(n) = X, then $T(2n) = X^3$ (or equivalently if T(n) = O(g(n)) then prove that $T(3n) = O(g(n)^3)$ [6 Marks]

1.d. For the following function f(n), find a simple function g(n) such that $f(n) = \Theta(g(n))$.

$$f(n) = (0.99)^n + n^{100}$$ [3 Marks]

1.e. An array A contain n-1 unique integers in the range [0,n-1], that is, there is one number from this range that is not in A. Design an O(n)–time algorithm for finding that number. You are allowed to use only O(1) additional space besides the array A itself. [6 Marks]

2.a. Determine θ for the following code fragments in the average case. Assume that all variables are of type **int**. [2 + 4 + 4 Marks]

(i)
```
sum = 0;
for (i=0; i<3; i++)
for (j=0; j<n; j++)
sum++;
```

(ii) Assume that array A contains n values, **Random** takes constant time, and **sort** takes n log n steps.
```
for (i=0; i<n; i++) {
for (j=0; j<n; j++)
A[i] = Random(n);
sort(A, n);
}
```

(iii)
```
sum = 0;
if (EVEN(n))
for (i=0; i<n; i++)
sum++;
else
sum = sum + n;
```

2.b. Given an array storing integers ordered by value, modify the binary search routine to return the position of the integer with the greatest value less than K when K itself does not appear in the array. Return **ERROR** if the least value in the array is greater than K. [5 Marks]

2.c. Prove that any comparison based algorithm requires Ω(n logn) comparisons in the worst case. [5 Marks]

[PTO]

3.a. Devise a comparison based sorting algorithm on a sequence of n¾ elements that takes O(n) time in worst case. [6 Marks]

3.b. Consider the following variation on Merge sort for large values of n. Instead of recursing until n is sufficiently small, recur at most a constant r times, and then use insertion sort to solve the 2^r resulting sub-problems. What is the (asymptotic) running time of this variation as a function of n? [8 Marks]

3.c. Let A[1..n] be an array such that the first $n - n^{1/2}$ elements are already sorted (though we know nothing about the remaining elements). Give an algorithm that sorts A in substantially better than n log n steps. [6 Marks]