Master of Computer Applications
Department of Computer Science & Engineering,
Motilal Nehru National Institute of Technology Allahabad.

End Semester Examination 2016-17 (odd)

Subject: Analysis of Algorithms (CA 3304) **Duration:** 3 HRS **Max. Marks:** 60

Note: Be specific and **to the point** in your answers. Make assumptions wherever necessary and quote it. All questions are compulsory. Neat answers, without cuttings will be appreciated. It is advisable to **design the solution in rough before writing the final answer**. Answer the questions **serially**.

Q1. *k*-WAY MERGE algorithm or MULTIWAY MERGES are a specific type of sequence merge algorithms that specialize in taking in multiple sorted lists and merging them into a single sorted list. These merge algorithms take in a number of sorted lists greater than two. In the context of *k*-WAY MERGE design a fast algorithm for merging *k* sorted lists with complexity analysis. (Do not write code, write only steps with example). **[10]**

Q2. Write an algorithm to find the second smallest or second largest of n elements in $n + \lceil \log n \rceil - 2$ comparisons. Support your answer with examples. **[10]**

Q3. Prove that the time complexity of function *fun()* is $\Theta(nlgn)$. ***[10]***

```
void fun()
{ for (int i=1; i<=n; i++)
    for (int j=1; j<=log(i); j++)
      printf("LIFE IS SHORT  ENJOY IT");
}
```

Q4. Although merge sort runs in $\Theta(n\log_2 n)$ worst-case running time and insertion sort runs in $\Theta(n^2)$ worst-case time, the constant factors in insertion sort can make it faster in practice for small problem sizes on many machines. Thus, it makes sense to coarsen the leaves of the recursion by using insertion sort within merge sort when sub-problems become sufficiently small ($\leq k$). Consider a modification to merge sort in which n/k sub-lists of length k are sorted using insertion sort and then merged using the *k*-WAY MERGE algorithm, where k is a value to be determined by the developer. In context of the above scenario answer the following questions:

a) Show that insertion sort can sort the n/k sublists, each of length k, in $\Theta(nk)$ worst-case time. **[3]**
b) Write the time complexities of modified algorithm and standard merge sort. **[3]**
c) What is the largest value of k as a function of n, for which the modified algorithm has the same running time as standard merge sort, in terms of Θ notation? **[3]**
d) Let us assume that for inputs of size n, insertion sort runs at $8n^2$ while merge sort runs at $64n\lg_2 n$. As a developer find the values of n for which insertion sort beat merge sort? **[3]**
e) What accounts for the difference in worst case running times of merge sort and quick sort? **[3]**

Q5. Consider a *2-D* map with a horizontal river passing through its center. There are n cities on the southern bank with x-coordinates $a(1)$ ... $a(n)$ and n cities on the northern bank with x-coordinates $b(1)$ ... $b(n)$. You want to connect as many *north-south* pairs of cities as possible with bridges such that **no two bridges cross**. When connecting cities, you can only connect city i on the northern bank to city i on the southern bank. Propose a dynamic programming solution (recursive structure) for finding the maximum number of bridges that can be built. Also find the solution for sample data. **[15]**

Sample data:
Cities you can connect via bridges are:
(8,1), (1,2), (4,3), (3,4), (5,5), (2,6), (6,7), (7,8).
However, you cannot build bridge (4,3) and (3,4) simultaneously because they will cross. In the same way (3,4) and (2,6) cannot be constructed together. Many other cases like these may exist.