The LNM Institute of Information Technology, Jaipur
II Mid Term Examination – 2013
Data Structure and Algorithms

Time: 3 Hrs MM: 50

1. Determine Big-O for the following code fragments. Assume that all variables are of type int.
   i. 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);
}
```

   ii. Assume array **A** contains a random permutation of the values from 0 to $n - 1$.

```
sum = 0;
for (i=0; i<n; i++)
for (j=0; A[j]!=i; j++)
sum++;
```

   iii.

```
sum = 0;
for (i=0; i<n; i++)
            for(j=0; j<n; j*=2)
            sum++;
```

   iv.

```
sum = 0;
for(i = 0; i< n; i*=2)
for(j = 0; j < n; j++)
sum++;
```

(2 marks)

2. Using the definitions of big-Oh and Ω, find the upper and lower bounds for the following expressions. Be sure to state appropriate values for c and n_0.
   (a) $c_1 n^3 + c_2$
   (b) $c_3 n \log n + c_4 n$ (2 marks)

3. Representation of symmetric matrix has a space complexity of $O(n^2)$. How can you represent the symmetric matrix such that the implementation in O(n). Give the store and retrieve operations for the same. (3 marks)

4. Write a recursive algorithm to compute the value of the recurrence relation
   T(n) = T(n - 1) + T(n-2);
   T(1) = 1. (2 marks)

5. A common problem for compilers and text editors is to determine if the parentheses in a string are balanced and properly nested. Give an algorithm that returns the position in the string of the first offending paranthesis if the string is not properly nested and balanced.
   (4 marks)

6. Implement a function in C/C++ to merge two sorted linked list. The output list should also be sorted. Your algorithm should run in linear time on the length of the output list. (4 marks)

7. Imagine that you are designing an application where you need to perform the operations **Insert, Delete_Maximum, and Delete_Minimum**. For this application, the cost of inserting

The LNM Institute of Information Technology, Jaipur
II Mid Term Examination – 2013
Data Structure and Algorithms

Time: 3 Hrs MM: 50

is not important because it can be done off-line prior to the startup of the time critical section, but the performance of the two deletion operations are critical. Repeated deletions of the either kind must work as fast as possible. Suggest a data structure that can support this application, and justify your suggestion. What is the time complexity for each of the three key operations? (4 marks)

8. Show the *min-heap* that results from running **buildHeap** on the following values stored in an array: 10, 5, 12, 3, 2, 1, 8, 7, 9, 4.
   Where in a max-heap might the smallest element reside? (4 marks)

9. A deque is like a queue, except that items may be added and removed from both the front and the rear. Write an array beased implementation for the deque. (4 marks)

10. Using closed hashing, with double hashing to resolve collisions, insert the following keys into a hash table of thirteen slots (the slots arew numbered 0 through 12). The hash functions to be used are H1 and H2, defines below. You should show the hash table after all the eight keys have been inserted. Indicate how you are using H1 and H2 to do the hashing. Function **Rev(*k*)** reverses the decinal digits of *k*, for example, Rev(37) = 73; Rev(7) = 7.
    H1(k) = k mod 13; H2(k) = (Rev(k+1) mod 11).
    Keys: 2, 8, 31, 20, 19, 18, 53, 27. (4 marks)

11. Write an algorithm that takes as imput the pointer to the root of a binary tree and prints the node values of the tree in level order. Level order first prints the root, then all nodes of level 1, then all nodes of level 2, and so on. (4 marks)

12. Describe an implementation of Depth First Search (DFS) on an undirected graph *G(V, E)*. What is the worst case complexity of your algorithm. Give three applications of DFS. (4 marks)

13. Prove that height of an AVL tree is O(log n). Show the result of inserting 2, 1, 4, 5, 9, 3, 6, 7 into an initially empty AVL tree. (4 marks)

14. Write an Insertion Sort algorithm for integer key values. When implementing Insertion sort, a binary search could be used to locate the position within the first $i - 1$ elements of the array into which element i should be inserted. How would this affect the number of comparisions required? How would using such a binary search affect the asymptotic running time for Insertion Sort. (5 marks)