DATA STRUCTURE AND BASIC ALGORITHMS
(CSE2004)

Time Allotted : 2½ hrs **Full Marks : 60**

Figures out of the right margin indicate full marks.

Candidates are required to answer Group A and any 4 (four) from Group B to E, taking one from each group.

Candidates are required to give answer in their own words as far as practicable.

Group – A

1. Answer any twelve: **12 × 1 = 12**

Choose the correct alternative for the following

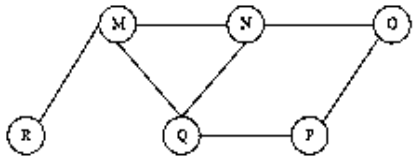
(i) What is the time complexity of a program to reverse a linked list?
(a) O(1) (b) O(n)
(c) $O(n^2)$ (d) None of the above

(ii) Let A be a two-dimensional array declared as follows:
A: array [1…10] [1…15] of character;
Assuming that each character takes one memory locations the array is stored in row-major order and the first element of the array is stored in location 100, what is the address of the element A[i][j] ?
(a) 15i + j + 84 (b) 15j + i + 84
(c) 10i + j + 89 (d) 10j + i + 89

(iii) Which of the following operations is not possible directly on a stack?
(a) Push (b) Pop
(c) Peek (d) Search

(iv) The equivalent Postfix notation for ((A + B) * C – (D – E) ^ (F + G)) is:
(a) AB + CDE – FG+^–* (b) ^ - * +ABC - DE + FG
(c) (AB)+CDE* -- FG+^ (d) AB*+DE--FG+^

(v) The Breadth First Search algorithm has been implemented using the queue data structure. One possible order of visiting the nodes of the following graph is



(a) MNOPQR (b) NQMPOR
(c) QMNPRO (d) QMNPOR

(vi) What is the worst case time complexity of inserting a node in a binary tree having 'n' nodes?
(a) O(n) (b) $O(n^2)$
(c) O(1) (d) $O(\log_2 n)$

(vii) What is the worst case time complexity of deleting a node in an AVL tree having 'n' nodes?
(a) O(n) (b) $O(n^2)$
(c) $O(\log_2 n)$ (d) O(1)

(viii) A hash function f is defined as f(key)= key mod 7, with a linear probing insert the keys 37, 38, 72, 48, 98, 11, 56, into a table indexed from 0, in which location the key 11 will be stored (Count table index 0 as 0^{th} location)?
(a) 1 (b) 2
(c) 5 (d) 6

(ix) Which of the following is true about binary search?
(a) It can be applied to any list, sorted or unsorted.
(b) It has a worst-case time complexity of O(n).
(c) It requires the list to be sorted before the search is performed.
(d) It works faster than linear search on all types of lists.

(x) When does the worst case complexity happen for Insertion Sort?
(a) When the array is already sorted
(b) When the array is sorted in reverse order
(c) When the array is not sorted
(d) When the array is sorted till the midpoint and then reverse sorted

Fill in the blanks with the correct word

(xi) In a/an _____ list, the last node has a pointer to the first node.

(xii) In a queue, elements are inserted at the _____ and removed from the _____.

(xiii) The minimum number of nodes in a binary heap of height 'h' is _____.

(xiv) In _____ the key is separated into different groups to generate the hash value.

(xv) The lower limit is modified when the key is _____ the middle element in the array in a binary search method.

Group - B

2. (a) Consider a two-dimensional integer array of M with 4 rows and 3 columns. Assume 5000 is the starting address of 2D array M. Now evaluate M[2][2] address using column-major formula. *[(CO1)(Understand/IOCQ)]*

(b) Consider the following loop and calculate its time complexity.

```
for(i=1;i<=n; i++)
{
    printf("Hello world\n");
    for(j=1;j<=i; j++)
        for(k=1;k<=i; k++)
            printf("Hello world of C\n");
}
```

[(CO1)(Evaluate/HOCQ)]

(c) Calculate the time complexity of the following loop.

```
while(n!=1)
{
    n=n/2;
    printf("Data structure\n");
}
```

[(CO1)(Evaluate/HOCQ)]

4 + 4 + 4 = 12

3. (a) What is the time complexity to insert an element into sorted array? *[(CO1)(Analyse/IOCQ)]*

(b) Consider two different polynomials $X_1$and X_2. Write the C code to add two polynomials into X_3. *[(CO2)(Apply/HOCQ)]*

(c) What is the time complexity of above mention code? *[(CO1)(Analyse/IOCQ)]*

2 + 8 + 2 = 12

Group - C

4. (a) Convert the following infix to postfix notation using STACK: K+L-M*N+(O^P)*W/U/V*T+Q. *[(CO3)(Apply/IOCQ)]*

(b) Explain how your algorithm handles operator precedence and parentheses. *[(CO1)(Remember/HOCQ)]*

(c) What is an abstract data type? Explain with examples. *[(CO2)(Understand/LOCQ)]*

6 + 4 + 2 = 12

5. (a) Explain the Tower of Hanoi problem with the algorithm. How it can be solved using recursion () [trace for 3 discs]. *[(CO2)(Understand/IOCQ)]*

(b) Explain how the problem of finding factorial can be solved using a recursive function (using both direct and indirect recursion). *[(CO2)(Understand/LOCQ)]*

(3 + 3) + (3 + 3) = 12

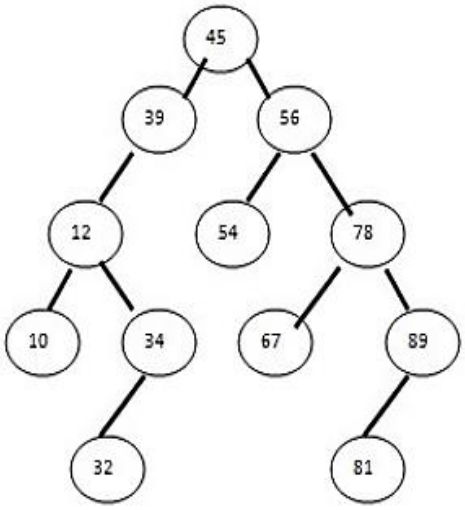
Group - D

6. (a) The structure of a node of a binary tree is defined in C as follows:-

```
typedef struct Node
{
int val;
struct Node *left, *right;
} tree;
```

Given the pointer to the root of such a binary tree of 'n' nodes write functions for the following traversals which take the root pointer as argument & mention the time complexity of each of the algorithms:- i) Inorder ii)Preorder iii)Postorder *[(CO3)(Analyse/LOCQ)]*

(b) Obtain the preorder, inorder and postorder traversals of the following binary search tree.



[(CO3)(Apply/IOCQ)]

(2 + 2 + 2) + 6 = 12

7. (a) Write down the differences between a binary search tree and an AVL tree using appropriate points of differences. *[(CO3)(Analyse/LOCQ)]*

   (b) Create an AVL tree with the input sequence: 96, 3, 43.
   (i) Now insert 27 and 35 into the created tree.
   (ii) Delete the value 35. *[(CO6)(Apply/HOCQ)]*

4 + (4 + 1 + 2 + 1) = 12

Group - E

8. (a) Illustrate the working of Heap Sort. *[(CO2)(Understand/LOCQ)]*

   (b) Show step by step construction of a max heap from the given array:
   9, 18, 7, 26, 2,14, 3, 30, 1. *[(CO2)(Understand/IOCQ)]*

   (c) Compare Heap Sort with other sorting algorithms in terms of time complexity. *[(CO4)(Analyse/HOCQ)]*

4 + 4 + 4 = 12

9. (a) Given the following sequence of elements insert these into a hash table of size 7 & use linear probing to resolve collisions (if any):
   73, 93, 40, 47, 10, 55, 63
   Use the hash function (h(k) + i) mod 7, h(k)= k mod 7, i Є {0,1,2,...,6}. *[(CO1)(Apply/IOCQ)]*

   (b) Write a recursive function to implement binary search for an integer element 'x' in an integer array of size 'n'. *[(CO4)(Apply/LOCQ)]*

   (c) Analyze the time complexity of the binary search algorithm using recurrence relation. *[(CO5)(Analyse/IOCQ)]*

4 + 4 + 4 = 12

Cognition Level	LOCQ	IOCQ	HOCQ
Percentage distribution	27.08	39.58	33.34