Course Code	**:**	**MCS-208**
Course Title	**:**	**Data Structures and Algorithms**
Assignment Number	**:**	**PGDCA_NEW(II)/208/Assign/2026-27**
Maximum Marks	**:**	**100**
Weightage	**:**	**30%**
Last Date for Submission	**:**	**31th October 2026 (for July, 2026 Session)** **15th April, 2027 (for January, 2027 Session)**

There are four questions in this assignment, which carry 80 marks. Each question carries 20 marks. Rest 20 marks are for viva voce. All algorithms should be written nearer to C programming language. You may use illustrations and diagrams to enhance the explanation, if necessary. Please go through the guidelines regarding assignments given in the Programme Guide for the format of presentation.

Q1: Write an algorithm to list the data fields of the nodes of a binary tree by level. Within levels, nodes are to be listed left to right. **(20 Marks)**

Q2: Show that every tree is a bipartite graph. **(20 Marks)**

Q3: Consider a hypothetical data object X1. X1 is a linear list with the restriction that while additions to the list may be made at either end, deletions can be made from one end only. Design a linked list representation for X1. Write addition and deletion algorithms for X1. Specify initial and boundary conditions for your representation. **(20 Marks)**

Q4: Write an algorithm to list all the identifiers in a hash table in lexicographic order. Assume the hash function f is f(X) = first character of X and linear probing is used. How much time does your algorithm take? **(20 Marks)**

Q1: Write an algorithm to list the data fields of the nodes of a binary tree by level. Within levels, nodes are to be listed left to right. (20 Marks)

Answer

The required order is the level-order traversal of a binary tree. A queue is the natural supporting data structure because nodes must be processed in the same order in which they are discovered. The root is visited first. Whenever a node is removed from the front of the queue, its left child is inserted before its right child. Hence, when the next level is reached, the nodes of that level are already arranged from left to right.

Representation and basic idea

Assume that every tree node has three fields: data, left and right. The data field is the value to be listed. The child fields contain pointers to the left and right child respectively. A queue stores pointers to nodes that have been discovered but not yet processed. To show the levels clearly, the algorithm records the number of nodes currently present in the queue at the beginning of each level. Exactly that many nodes are removed and printed before moving to the next level.

```
typedef struct node {
    int data;
    struct node *left;
    struct node *right;
} NODE;

/* Queue stores NODE pointers.  Assume the usual operations:
   initQueue, isEmpty, enqueue, dequeue and queueSize. */

void listByLevel(NODE *root)
{
    QUEUE q;
    NODE *p;
    int count, i;

    if (root == NULL)                /* empty tree */
        return;

    initQueue(&q);
    enqueue(&q, root);

    while (!isEmpty(&q)) {
        count = queueSize(&q);       /* nodes of the present level */

        for (i = 0; i < count; i++) {
            p = dequeue(&q);
            printf("%d ", p->data);
```