

**Data Structure and Algorithm (MCA 271)**

**ESE 3 –**

***BY***

**Himanshu Heda (24225013)**

**SUBMITTED TO**

**Prof. Vandna Kansal**

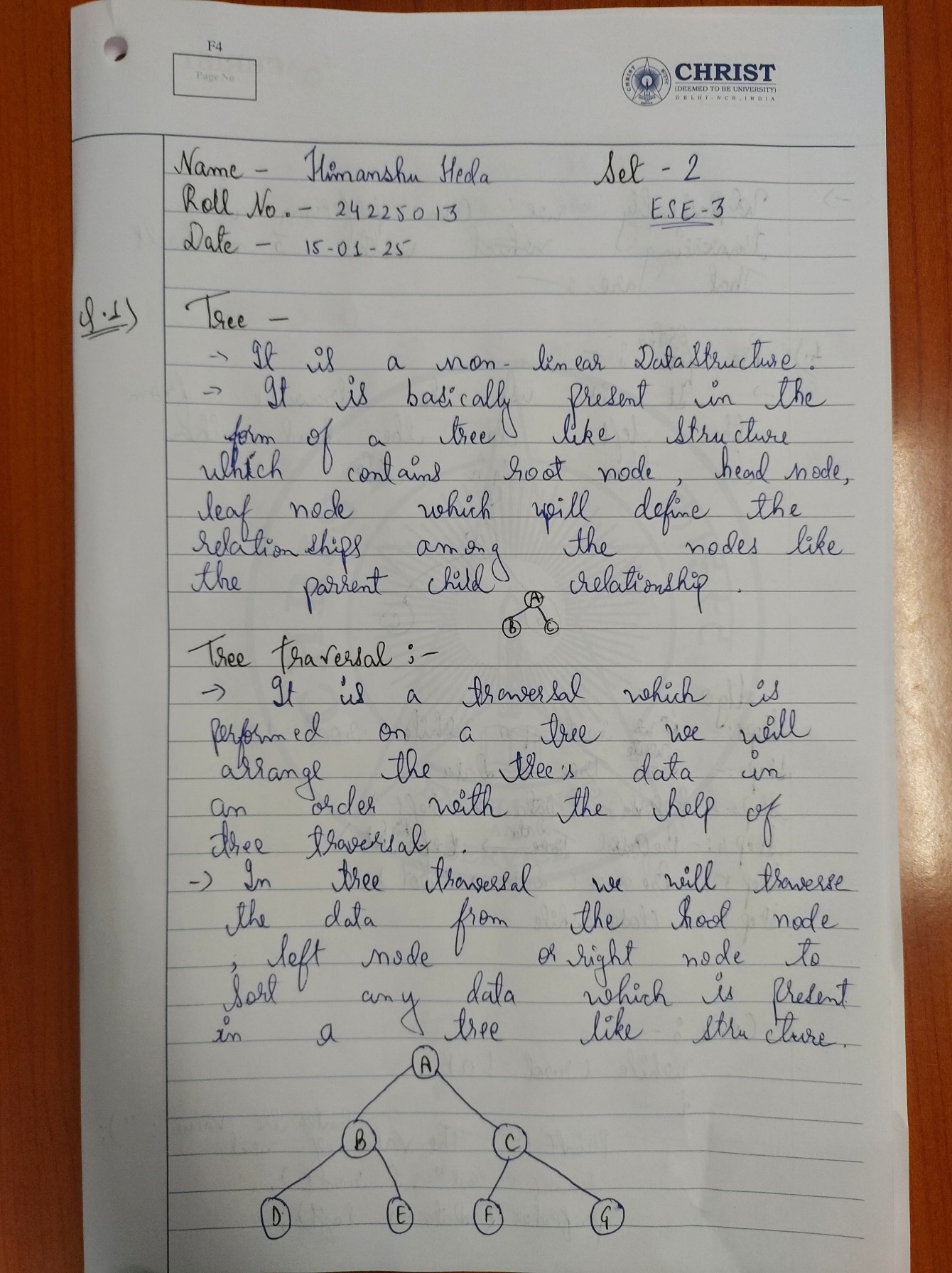
**SCHOOL OF SCIENCES**

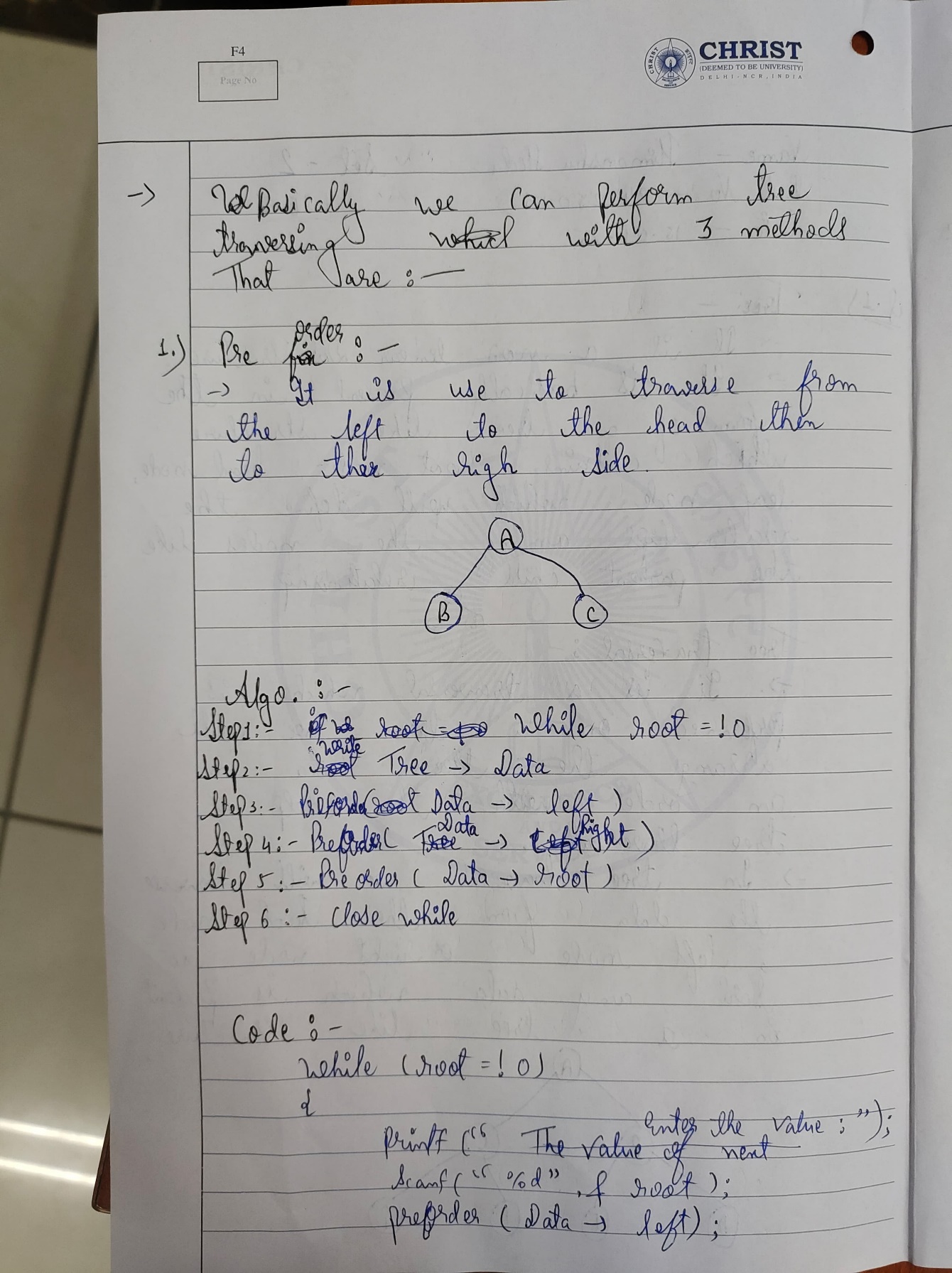
**2024-2025**

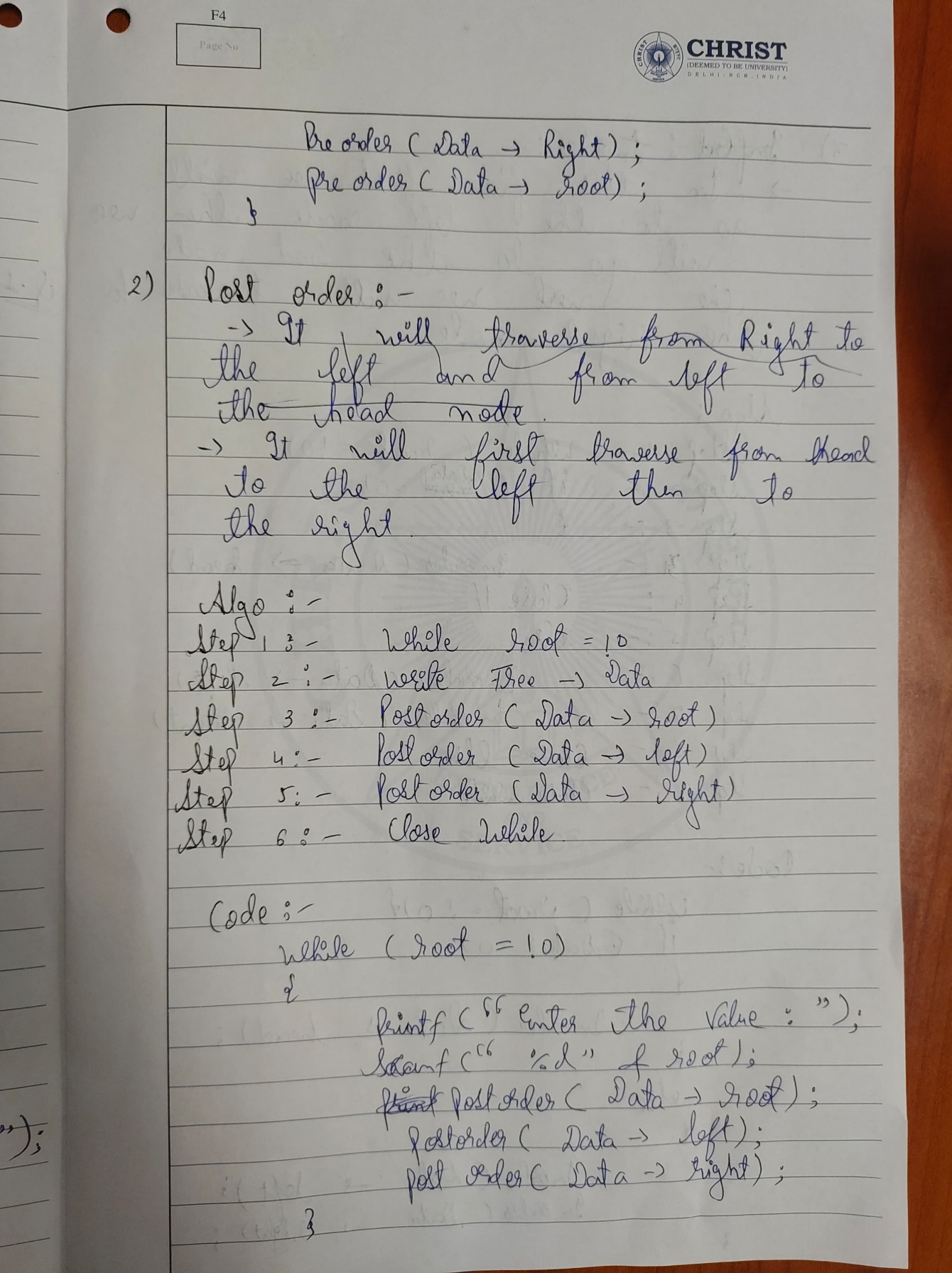
**Program Description:**

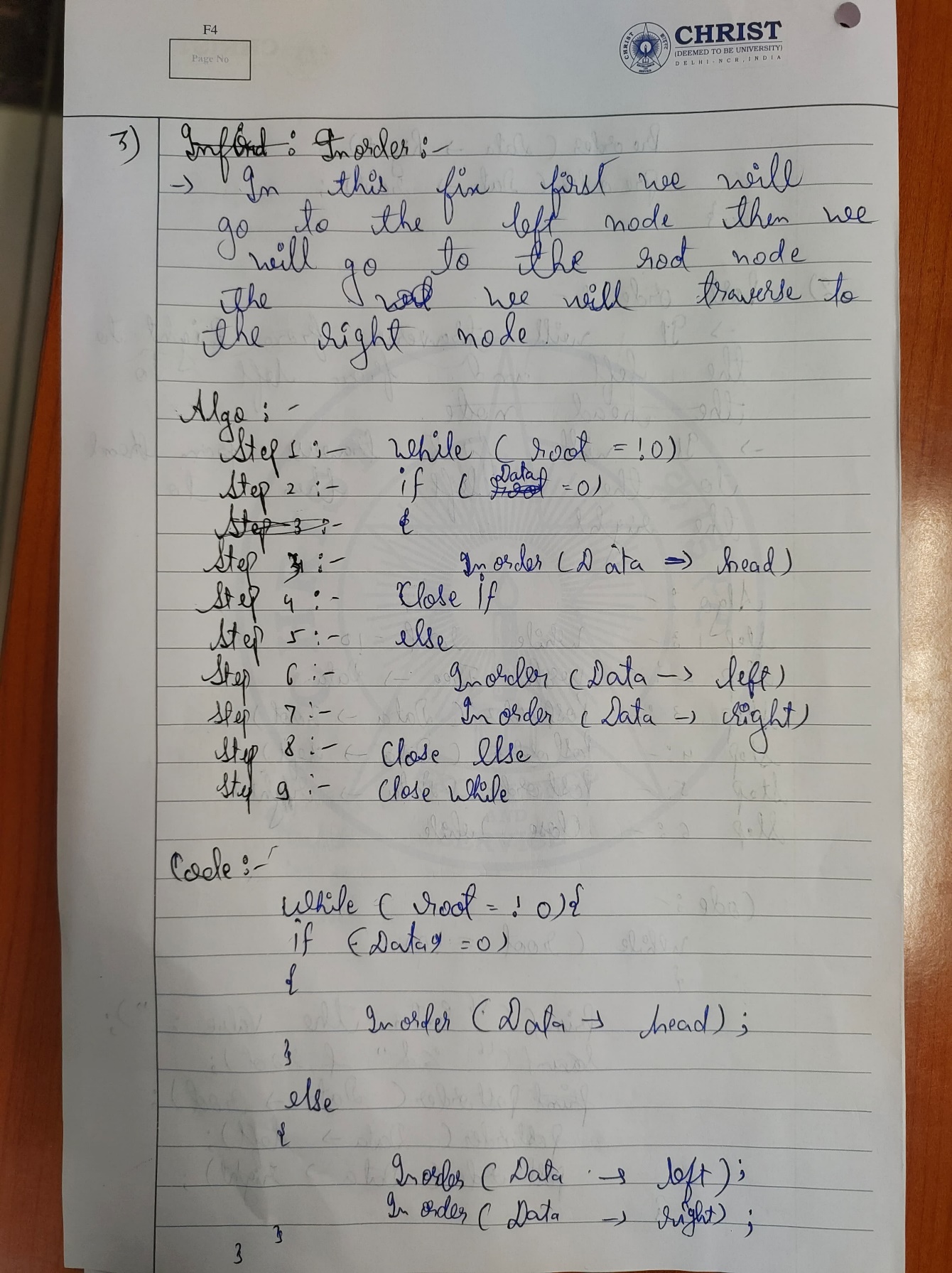
**Code of the program**

**Output**: - Paste the o/p of the program.









Tree Traversal : --

#include <stdio.h>

#include <stdlib.h>

// Define the structure for a binary tree node

struct Node {

    int data;

    struct Node\* left;

    struct Node\* right;

};

// Function to create a new node

struct Node\* createNode(int data) {

    struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));

    newNode->data = data;

    newNode->left = NULL;

    newNode->right = NULL;

    return newNode;

}

// Function to perform In-order traversal

void inOrderTraversal(struct Node\* root) {

    if (root == NULL) return;

    inOrderTraversal(root->left);

    printf("%d ", root->data);

    inOrderTraversal(root->right);

}

// Function to perform Pre-order traversal

void preOrderTraversal(struct Node\* root) {

    if (root == NULL) return;

    printf("%d ", root->data);

    preOrderTraversal(root->left);

    preOrderTraversal(root->right);

}

// Function to perform Post-order traversal

void postOrderTraversal(struct Node\* root) {

    if (root == NULL) return;

    postOrderTraversal(root->left);

    postOrderTraversal(root->right);

    printf("%d ", root->data);

}

int main() {

    // Creating a sample binary tree

    struct Node\* root = createNode(1);

    root->left = createNode(2);

    root->right = createNode(3);

    root->left->left = createNode(4);

    root->left->right = createNode(5);

    root->right->left = createNode(6);

    root->right->right = createNode(7);

    // Performing In-order traversal

    printf("In-order Traversal: ");

    inOrderTraversal(root);

    printf("\n");

    // Performing Pre-order traversal

    printf("Pre-order Traversal: ");

    preOrderTraversal(root);

    printf("\n");

    // Performing Post-order traversal

    printf("Post-order Traversal: ");

    postOrderTraversal(root);

    printf("\n");

    return 0;

}

OUTPUT : --

