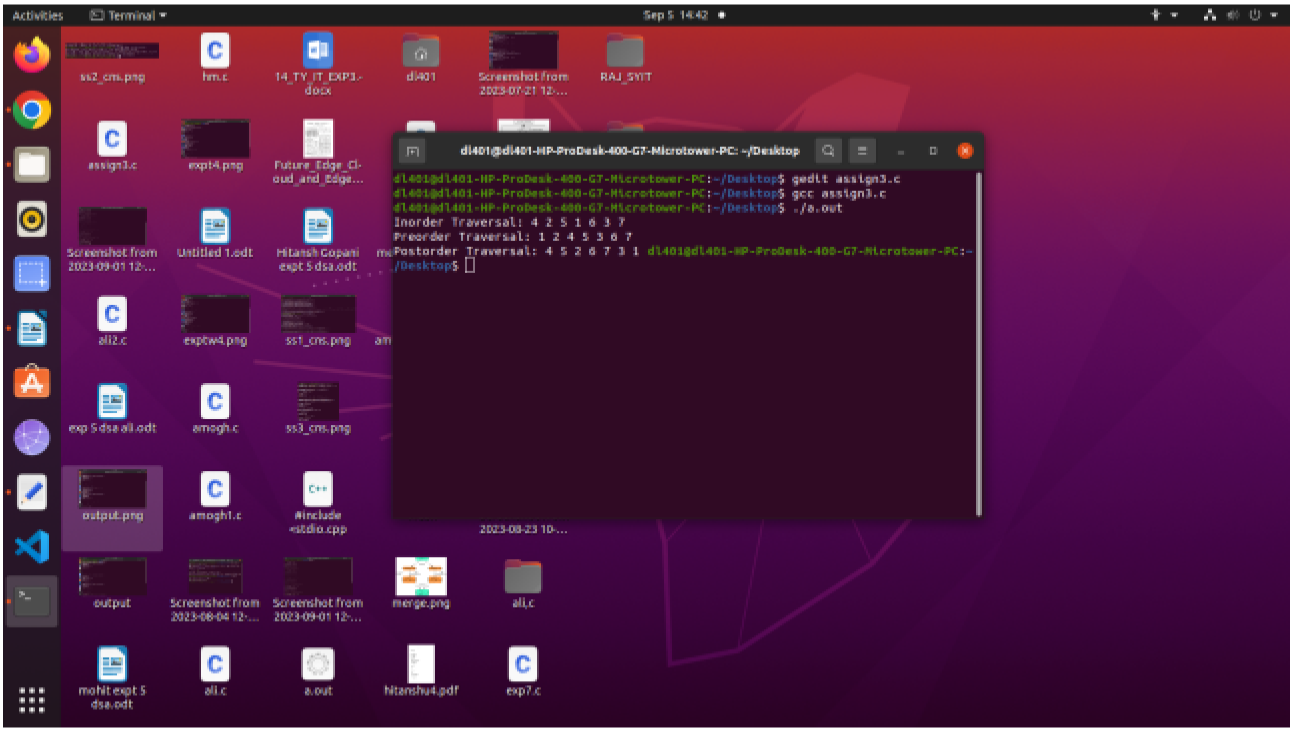
EXP 6

```
#include <stdio.h>
#include <stdlib.h>

// Define the structure for a binary tree node
struct TreeNode {
    int value;
    struct TreeNode* left;
    struct TreeNode* right;
};

// Function to create a new tree node
struct TreeNode* createNode(int value) {
    struct TreeNode* newNode = (struct TreeNode*)malloc(sizeof(struct TreeNode));
    if (newNode) {
        newNode->value = value;
        newNode->left = NULL;
        newNode->right = NULL;
    }
    return newNode;
}

// Function for inorder traversal
void inorderTraversal(struct TreeNode* node) {
    if (node) {
        inorderTraversal(node->left);
        printf("%d ", node->value);
        inorderTraversal(node->right);
    }
}

// Function for preorder traversal
void preorderTraversal(struct TreeNode* node) {
    if (node) {
        printf("%d ", node->value);
        preorderTraversal(node->left);
        preorderTraversal(node->right);
    }
}

// Function for postorder traversal
void postorderTraversal(struct TreeNode* node) {
    if (node) {
        postorderTraversal(node->left);
        postorderTraversal(node->right);
        printf("%d ", node->value);
    }
}

// Function to insert a new node into the binary tree
struct TreeNode* insertNode(struct TreeNode* root, int value) {
    if (root == NULL) {
        return createNode(value);
    }

    if (value < root->value) {
        root->left = insertNode(root->left, value);
    } else if (value > root->value) {
        root->right = insertNode(root->right, value);
    }

    return root;
}

int main() {
    struct TreeNode* root = NULL;
    int numNodes, value;

    printf("Enter the number of nodes in the binary tree: ");
    scanf("%d", &numNodes);

    printf("Enter the values of the nodes:\n");
    for (int i = 0; i < numNodes; i++) {
        scanf("%d", &value);
        root = insertNode(root, value);
    }

    printf("Inorder Traversal: ");
    inorderTraversal(root);

    printf("\nPreorder Traversal: ");
    preorderTraversal(root);

    printf("\nPostorder Traversal: ");
    postorderTraversal(root);

    // Free the allocated memory for the tree nodes
    // In a production program, you should clean up memory properly.
    return 0;
}
```



```
dl401@dl401-HP-ProDesk-400-G7-Microtower-PC:~/Desktop$ gedit assign3.c
dl401@dl401-HP-ProDesk-400-G7-Microtower-PC:~/Desktop$ gcc assign3.c
dl401@dl401-HP-ProDesk-400-G7-Microtower-PC:~/Desktop$ ./a.out
Inorder Traversal: 4 2 5 1 6 3 7
Preorder Traversal: 1 2 4 5 3 6 7
Postorder Traversal: 4 5 2 6 7 3 1 dl401@dl401-HP-ProDesk-400-G7-Microtower-PC:~
/Desktop$ 
```