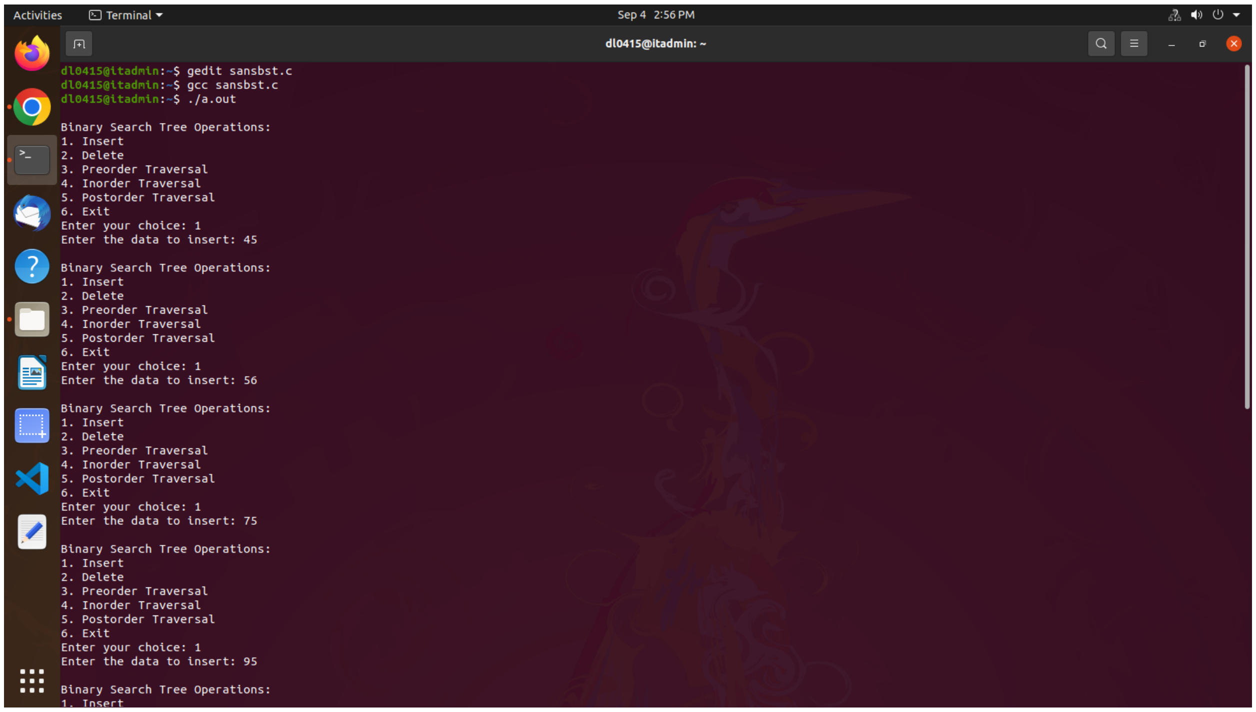
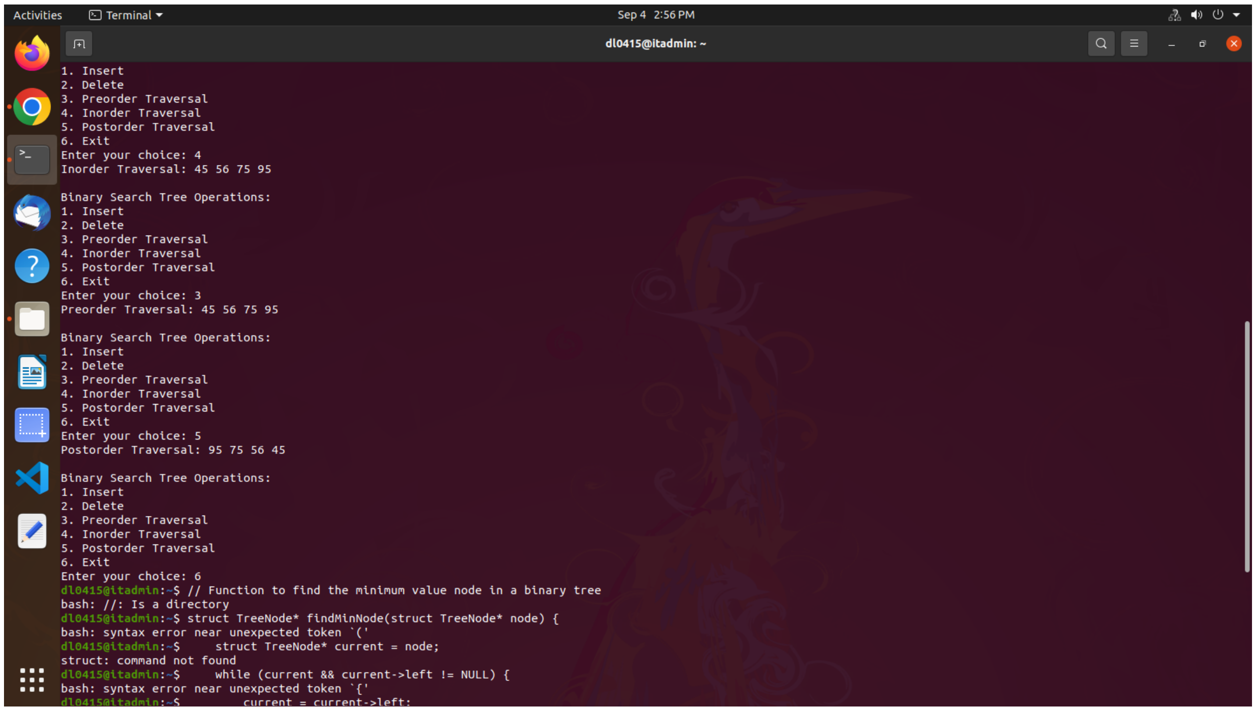
Assignment No.3
Sanskar Srivastava
SY-IT 57
CODE:-

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

struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed.\n");
        exit(1);
    }
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}

struct Node* insert(struct Node* root, int data) {
    if (root == NULL) {
        return createNode(data);
    }

    if (data < root->data) {
        root->left = insert(root->left, data);
    } else if (data > root->data) {
        root->right = insert(root->right, data);
    }

    return root;
}

struct Node* findMin(struct Node* node) {
    struct Node* current = node;
    while (current && current->left != NULL) {
        current = current->left;
    }
    return current;
}


struct Node* deleteNode(struct Node* root, int key) {
    if (root == NULL) {
        return root;
    }

    if (key < root->data) {
        root->left = deleteNode(root->left, key);
    } else if (key > root->data) {
        root->right = deleteNode(root->right, key);
    } else {
        if (root->left == NULL) {
            struct Node* temp = root->right;
            free(root);
            return temp;
        } else if (root->right == NULL) {
            struct Node* temp = root->left;
            free(root);
            return temp;
        }

        struct Node* temp = findMin(root->right);
        root->data = temp->data;
        root->right = deleteNode(root->right, temp->data);
    }

    return root;
}

// preorder traversal
void preorderTraversal(struct Node* root) {
    if (root == NULL) {
        return;
    }
    printf("%d ", root->data);
    preorderTraversal(root->left);
    preorderTraversal(root->right);
}) {
        printf("\nBinary Search Tree Operations:\n");
        printf("1. Insert\n");
        printf("2. Delete\n");
        printf("3. Preorder Traversal\n");
        printf("4. Inorder Traversal\n");
        printf("5. Postorder Traversal\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the data to insert: ");
                scanf("%d", &data);
                root = insert(root, data);
                break;
            case 2:
                printf("Enter the data to delete: ");
                scanf("%d", &data);
                root = deleteNode(root, data);
                break;
            case 3:
                printf("Preorder Traversal: ");
                preorderTraversal(root);
                printf("\n");
                break;
            case 4:
                printf("Inorder Traversal: ");
                inorderTraversal(root);
                printf("\n");
                break;
            case 5:
                printf("Postorder Traversal: ");
                postorderTraversal(root);
                printf("\n");
                break;
            case 6:
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }


// Function to perform inorder traversal
void inorderTraversal(struct Node* root) {
    if (root == NULL) {
        return;
```

```c
    }
    inorderTraversal(root->left);
    printf("%d ", root->data);
    inorderTraversal(root->right);
}

// Function to perform postorder traversal
void postorderTraversal(struct Node* root) {
    if (root == NULL) {
        return;
    }
    postorderTraversal(root->left);
    postorderTraversal(root->right);
    printf("%d ", root->data);
}

int main() {
    struct Node* root = NULL;
    int choice, data;

    while (1) {
        printf("\nBinary Search Tree Operations:\n");
        printf("1. Insert\n");
        printf("2. Delete\n");
        printf("3. Preorder Traversal\n");
        printf("4. Inorder Traversal\n");
        printf("5. Postorder Traversal\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter the data to insert: ");
                scanf("%d", &data);
                root = insert(root, data);
                break;
            case 2:
                printf("Enter the data to delete: ");
                scanf("%d", &data);
                root = deleteNode(root, data);
                break;
            case 3:
                printf("Preorder Traversal: ");
                preorderTraversal(root);
                printf("\n");
                break;
            case 4:
                printf("Inorder Traversal: ");
                inorderTraversal(root);
                printf("\n");
                break;
            case 5:
                printf("Postorder Traversal: ");
                postorderTraversal(root);
                printf("\n");
                break;
            case 6:
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}
```



```
dl0415@itadmin:~$ gedit sansbst.c
dl0415@itadmin:~$ gcc sansbst.c
dl0415@itadmin:~$ ./a.out

Binary Search Tree Operations:
1. Insert
2. Delete
3. Preorder Traversal
4. Inorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter the data to insert: 45

Binary Search Tree Operations:
1. Insert
2. Delete
3. Preorder Traversal
4. Inorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter the data to insert: 56

Binary Search Tree Operations:
1. Insert
2. Delete
3. Preorder Traversal
4. Inorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter the data to insert: 75

Binary Search Tree Operations:
1. Insert
2. Delete
3. Preorder Traversal
4. Inorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 1
Enter the data to insert: 95

Binary Search Tree Operations:
1. Insert
```



```
1. Insert
2. Delete
3. Preorder Traversal
4. Inorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 4
Inorder Traversal: 45 56 75 95

Binary Search Tree Operations:
1. Insert
2. Delete
3. Preorder Traversal
4. Inorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 3
Preorder Traversal: 45 56 75 95

Binary Search Tree Operations:
1. Insert
2. Delete
3. Preorder Traversal
4. Inorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 5
Postorder Traversal: 95 75 56 45

Binary Search Tree Operations:
1. Insert
2. Delete
3. Preorder Traversal
4. Inorder Traversal
5. Postorder Traversal
6. Exit
Enter your choice: 6
dl0415@itadmin:~$ // Function to find the minimum value node in a binary tree
bash: //: Is a directory
dl0415@itadmin:~$ struct TreeNode* findMinNode(struct TreeNode* node) {
bash: syntax error near unexpected token `('
dl0415@itadmin:~$     struct TreeNode* current = node;
struct: command not found
dl0415@itadmin:~$     while (current && current->left != NULL) {
bash: syntax error near unexpected token `{'
dl0415@itadmin:~$         current = current->left;
```