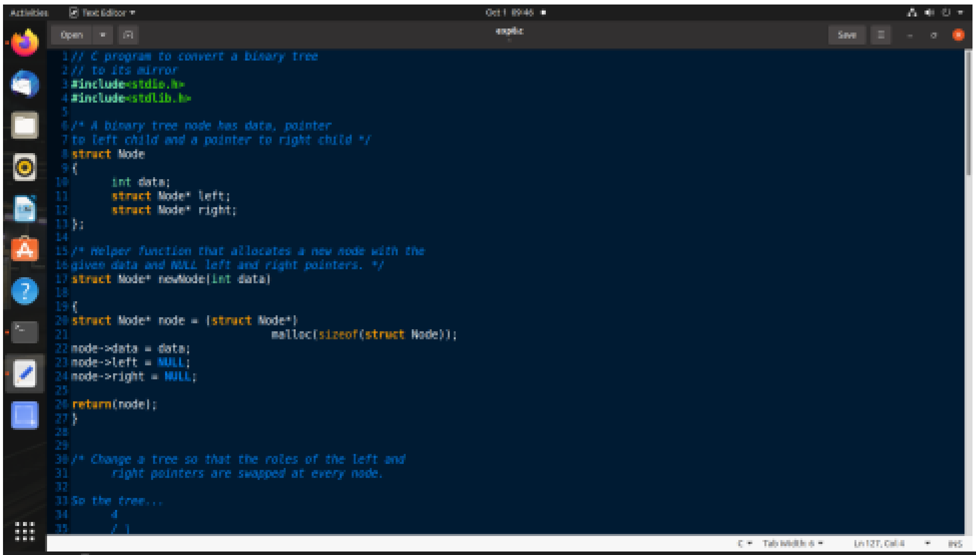
Aim:

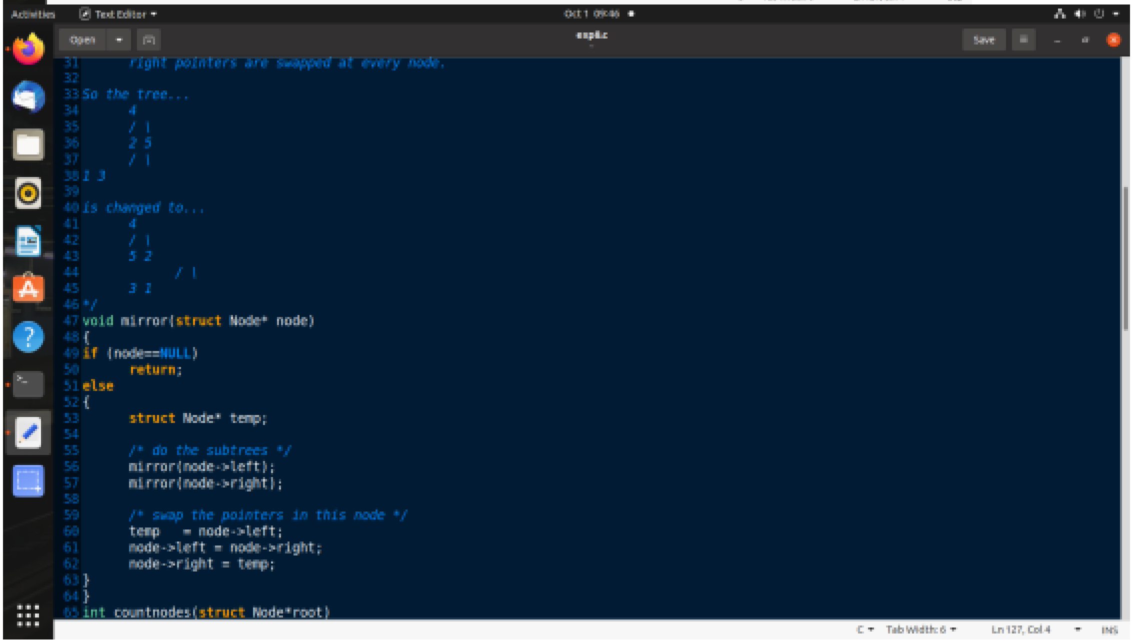
Experiment No. 08 (Implementation of various operations on tree like – copying tree, mirroring a tree,

counting the number of nodes in the tree, etc.)

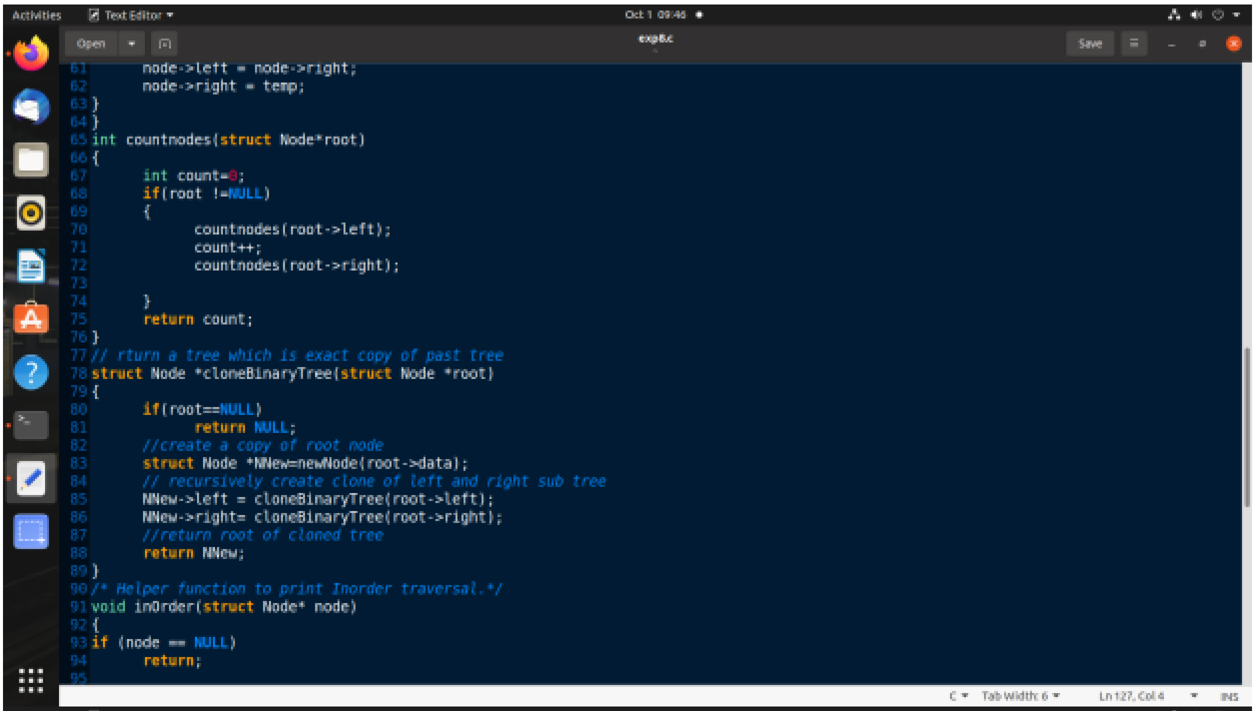
SOURCE CODE



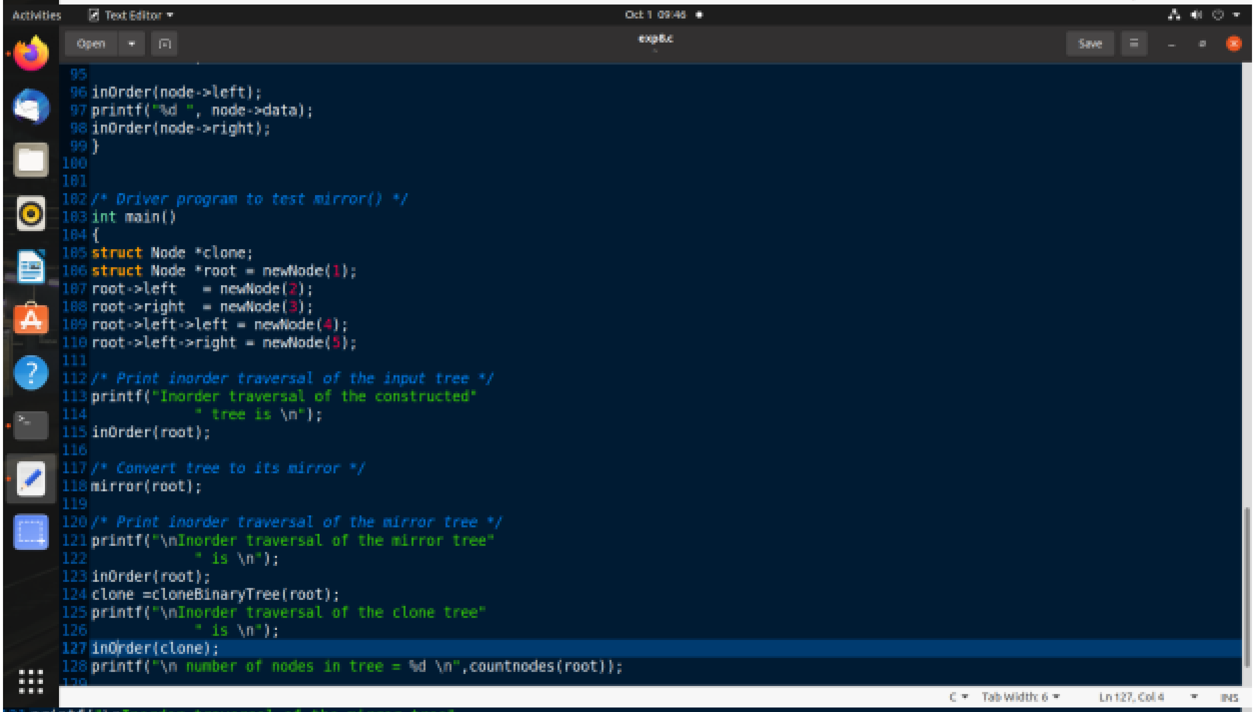
```c
// C program to convert a binary tree
// to its mirror
#include<stdio.h>
#include<stdlib.h>

/* A binary tree node has data, pointer
to left child and a pointer to right child */
struct Node
{
        int data;
        struct Node* left;
        struct Node* right;
};

/* Helper function that allocates a new node with the
given data and NULL left and right pointers. */
struct Node* newNode(int data)

{
struct Node* node = (struct Node*)
                                malloc(sizeof(struct Node));
node->data = data;
node->left = NULL;
node->right = NULL;

return(node);
}


/* Change a tree so that the roles of the left and
        right pointers are swapped at every node.

So the tree...
        4
        / \
```



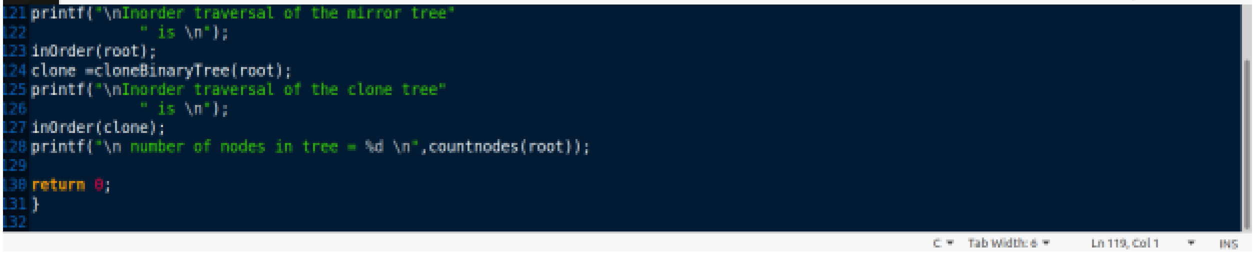
```c
        right pointers are swapped at every node.

So the tree...
        4
        / \
        2 5
        / \
1 3

is changed to...
        4
        / \
        5 2
                / \
        3 1
*/
void mirror(struct Node* node)
{
if (node==NULL)
        return;
else
{
        struct Node* temp;

        /* do the subtrees */
        mirror(node->left);
        mirror(node->right);

        /* swap the pointers in this node */
        temp    = node->left;
        node->left = node->right;
        node->right = temp;
}
}
int countnodes(struct Node*root)
```



```
        node->left = node->right;
        node->right = temp;
}
}
int countnodes(struct Node*root)
{
        int count=0;
        if(root !=NULL)
        {
                countnodes(root->left);
                count++;
                countnodes(root->right);

        }
        return count;
}
// rturn a tree which is exact copy of past tree
struct Node *cloneBinaryTree(struct Node *root)
{
        if(root==NULL)
                return NULL;
        //create a copy of root node
        struct Node *NNew=newNode(root->data);
        // recursively create clone of left and right sub tree
        NNew->left = cloneBinaryTree(root->left);
        NNew->right= cloneBinaryTree(root->right);
        //return root of cloned tree
        return NNew;
}
/* Helper function to print Inorder traversal.*/
void inOrder(struct Node* node)
{
if (node == NULL)
        return;
```



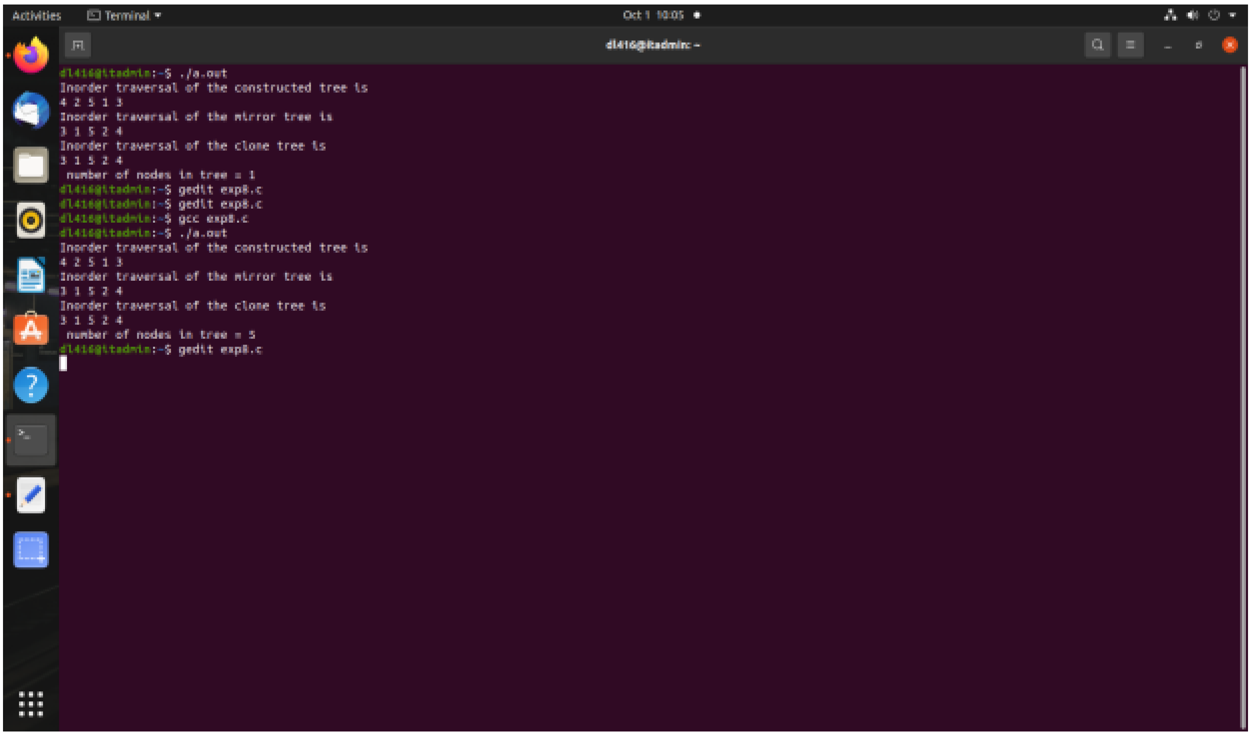
```

inOrder(node->left);
printf("%d ", node->data);
inOrder(node->right);
}


/* Driver program to test mirror() */
int main()
{
struct Node *clone;
struct Node *root = newNode(1);
root->left    = newNode(2);
root->right   = newNode(3);
root->left->left = newNode(4);
root->left->right = newNode(5);

/* Print inorder traversal of the input tree */
printf("Inorder traversal of the constructed"
            " tree is \n");
inOrder(root);

/* Convert tree to its mirror */
mirror(root);

/* Print inorder traversal of the mirror tree */
printf("\nInorder traversal of the mirror tree"
            " is \n");
inOrder(root);
clone =cloneBinaryTree(root);
printf("\nInorder traversal of the clone tree"
            " is \n");
inOrder(clone);
printf("\n number of nodes in tree = %d \n",countnodes(root));
```



```
printf("\nInorder traversal of the mirror tree"
            " is \n");
inOrder(root);
clone =cloneBinaryTree(root);
printf("\nInorder traversal of the clone tree"
            " is \n");
inOrder(clone);
printf("\n number of nodes in tree = %d \n",countnodes(root));
```

```

return 0;
}
```

OUTPUT



```
dl416@itadmin:~$ ./a.out
Inorder traversal of the constructed tree is
4 2 5 1 3
Inorder traversal of the mirror tree is
3 1 5 2 4
Inorder traversal of the clone tree is
3 1 5 2 4
 number of nodes in tree = 1
dl416@itadmin:~$ gedit exp8.c
dl416@itadmin:~$ gedit exp8.c
dl416@itadmin:~$ gcc exp8.c
dl416@itadmin:~$ ./a.out
Inorder traversal of the constructed tree is
4 2 5 1 3
Inorder traversal of the mirror tree is
3 1 5 2 4
Inorder traversal of the clone tree is
3 1 5 2 4
 number of nodes in tree = 5
dl416@itadmin:~$ gedit exp8.c
```