NAME:JAYKUMAR.P.GOR
ROLL NO.:16
SY-IT.
DSA LAB.

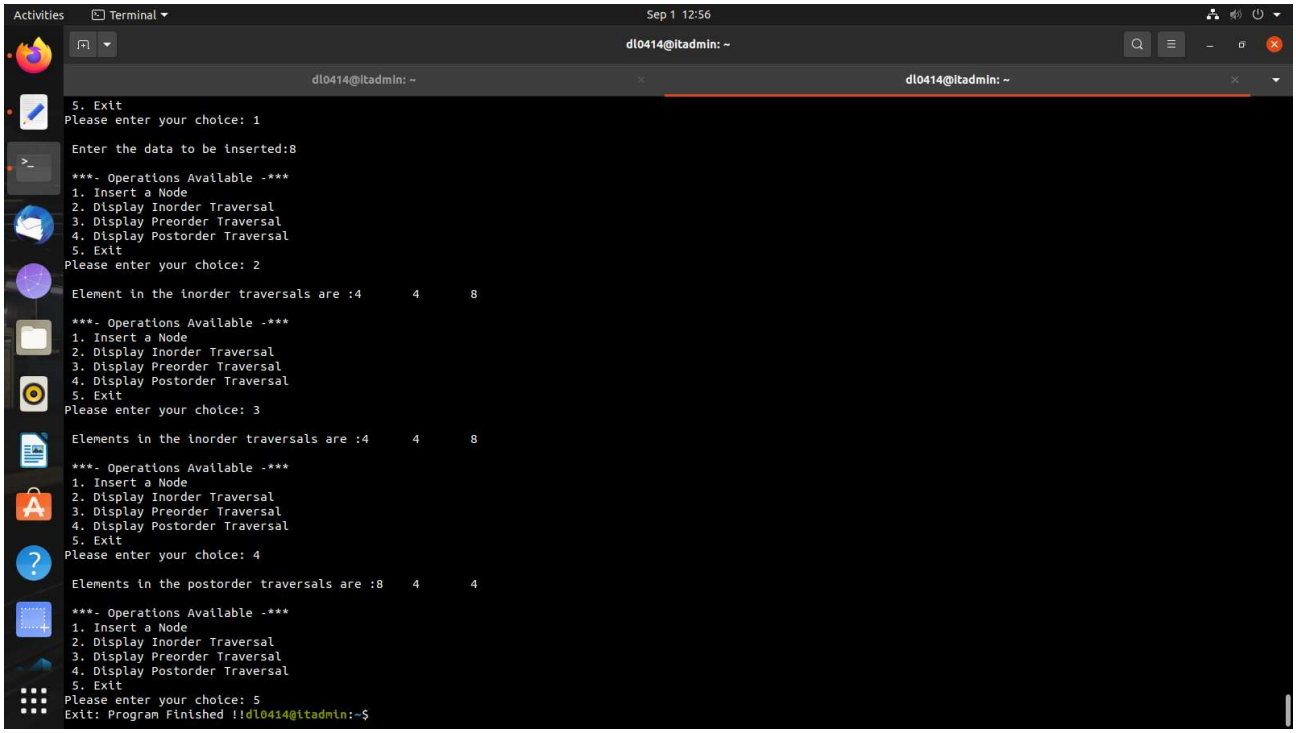
CODE:

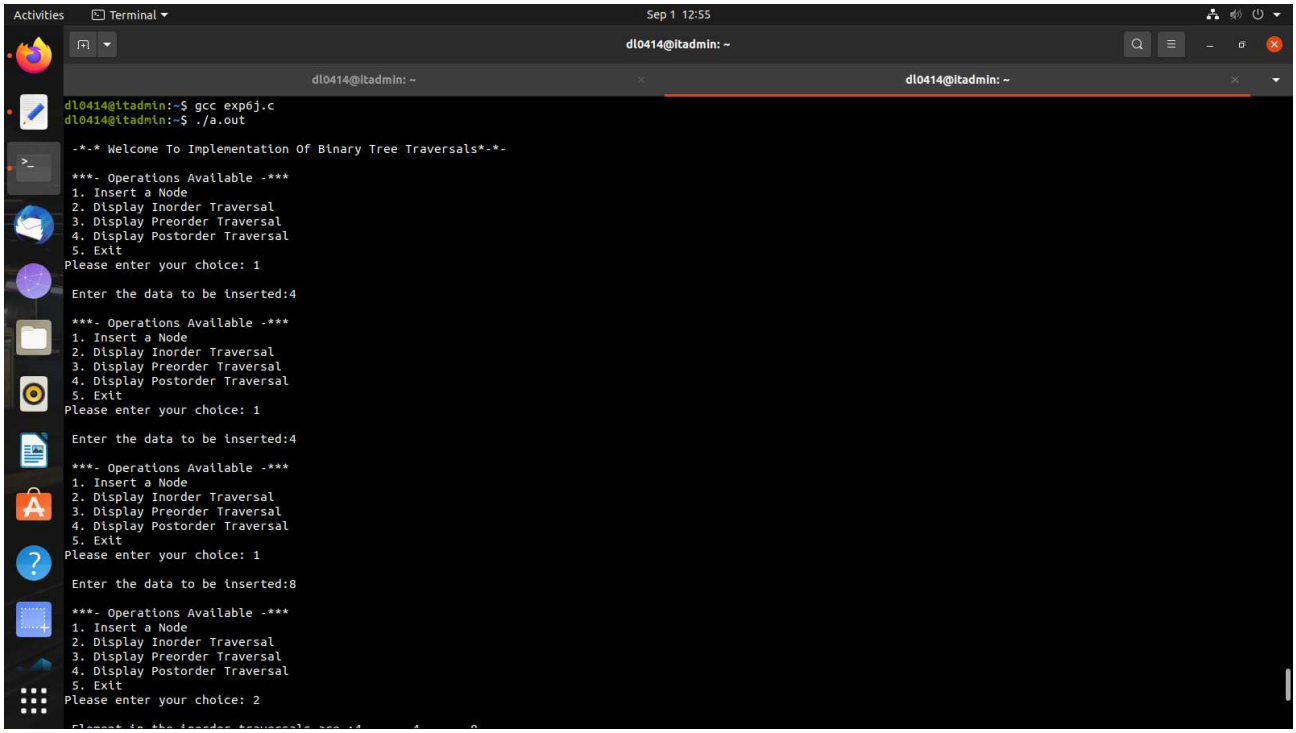
```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>


struct node
{
    int data;
    struct node *left;
    struct node *right;

};
struct node *tree;
void create(struct node *);
struct node *insert(struct node *,int);
void inorder(struct node *);
void preorder(struct node *);
void postorder(struct node *);

void main()
{
  printf("\n -*-* Welcome To Implementation Of Binary Tree Traversals*-*- \n");
  int choice,x;
  struct node *ptr;
  create(tree);
  do
  {
    printf("\n ***- Operations Available -*** ");
    printf("\n 1. Insert a Node");
    printf("\n 2. Display Inorder Traversal");
    printf("\n 3. Display Preorder Traversal");
    printf("\n 4. Display Postorder Traversal");
    printf("\n 5. Exit \n");
    printf("Please enter your choice: ");
    scanf("%d", &choice);
    switch (choice)
    {
    case 1:
    printf("\n Enter the data to be inserted:");
    scanf("%d",&x);
    tree = insert(tree,x);
    break;

    case 2:
    printf("\n Element in the inorder traversals are :");
    inorder(tree);
    printf("\n");
    break;

    case 3:
    printf("\n Elements in the inorder traversals are :");
    preorder(tree);
    printf("\n");
    break;
```

```
    case 4:
    printf("\n Elements in the postorder traversals are :");
    postorder(tree);
    printf("\n");
    break;

    case 5:
    printf("Exit: Program Finished !!");
    break;

    default:
    printf("\n Please enter a valid option 1,2,3,4,5.");
    break;
    }
  } while (choice != 5);
}

void create(struct node *tree)
{
  tree = NULL;
}

struct node *insert(struct  node *tree, int x)
{
   struct node *p, *temp, *root;
   p = (struct node *)malloc(sizeof(struct node));
   p->data = x;
   p->left = NULL;
   p->right = NULL;
   if (tree == NULL)
   {
   tree = p;
   tree->left = NULL;
   tree->right = NULL;
   }
   else
   {
   root = NULL;
   temp = tree;
   while (temp !=NULL)
   {
   root = temp;
   if (x < temp->data)
      temp = temp->left;
   else
     temp = temp->right;
   }
   if (x < root->data)
      root->left = p;
   else
      root->right = p;
   }
   return tree;
}

 void inorder(struct node *tree)
 {
   if (tree != NULL)
   {
      inorder(tree->left);
      printf("%d \t",tree->data);
      inorder(tree->right);
```

```
  }
}

void preorder(struct node *tree)
{
   if (tree != NULL)
   {
     printf("%d \t",tree->data);
     preorder(tree->left);
     preorder(tree->right);
   }
}

void postorder(struct node *tree)
{
  if (tree !=NULL)
  {
      postorder(tree->left);
      postorder(tree->right);
      printf("%d \t", tree->data);
  }
}
```

SCREENSHOT:




Activities
Terminal
Sep 1 12:55
dl0414@itadmin: ~
dl0414@itadmin: ~
dl0414@itadmin: ~
dl0414@itadmin:~$ gcc exp6j.c
dl0414@itadmin:~$ ./a.out
-*-* Welcome To Implementation Of Binary Tree Traversals*-*-
- Operations Available -
1. Insert a Node
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
5. Exit
Please enter your choice: 1
Enter the data to be inserted:4
- Operations Available -
1. Insert a Node
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
5. Exit
Please enter your choice: 1
Enter the data to be inserted:4
- Operations Available -
1. Insert a Node
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
5. Exit
Please enter your choice: 1
Enter the data to be inserted:8
- Operations Available -
1. Insert a Node
2. Display Inorder Traversal
3. Display Preorder Traversal
4. Display Postorder Traversal
5. Exit
Please enter your choice: 2