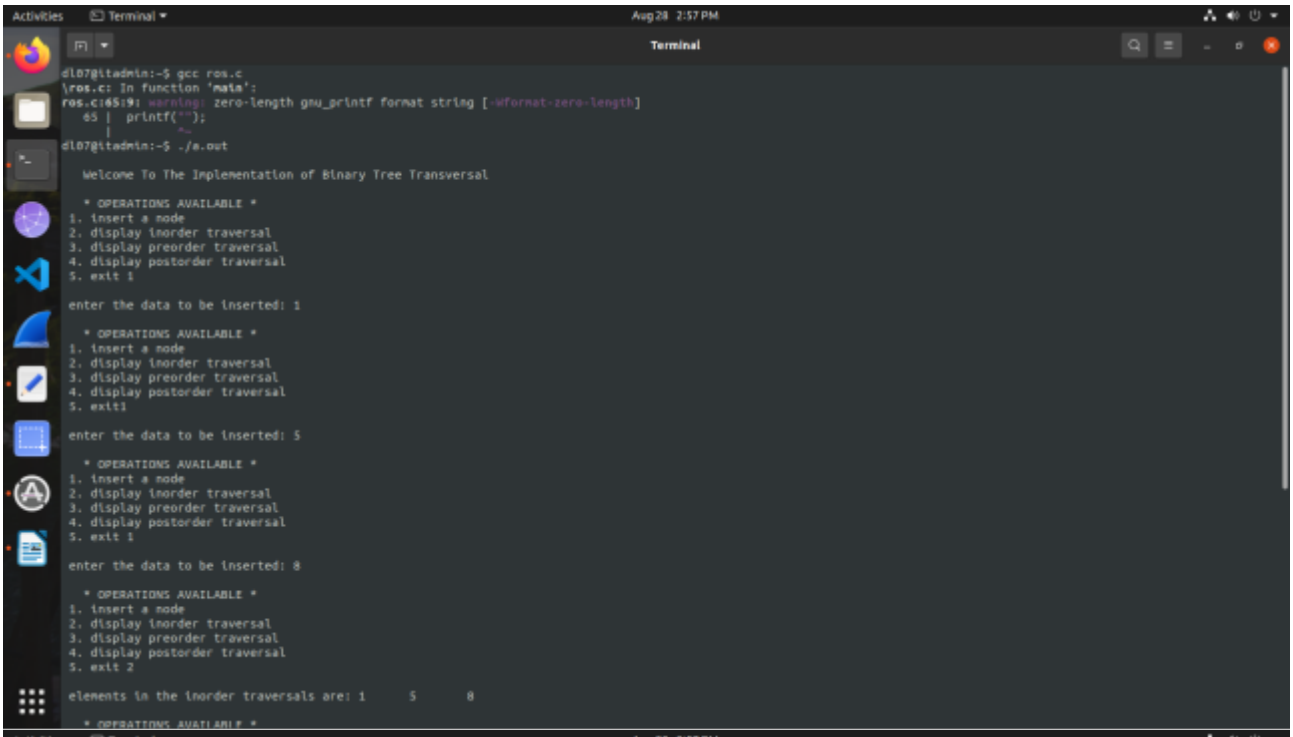
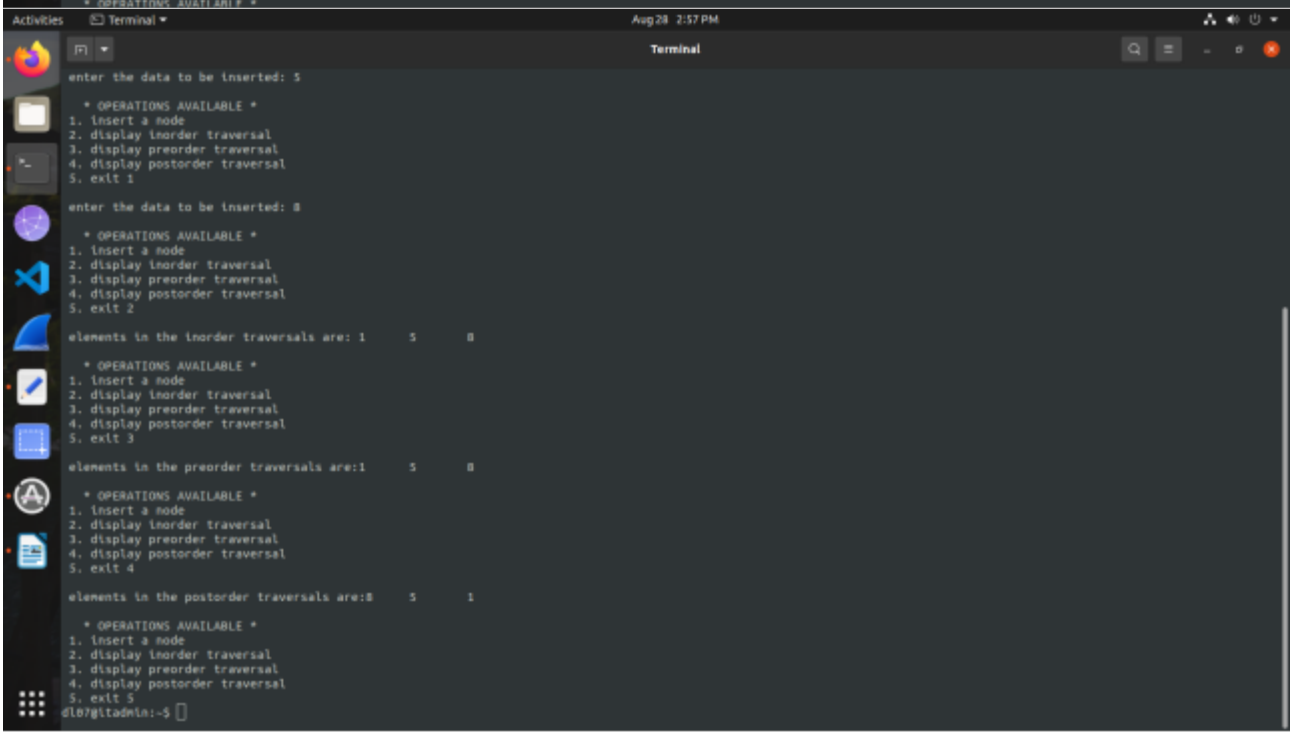
Roshan Rawat
SY-IT
48






```
#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 The Implementation of Binary Tree Transversal   \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");
      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 elements in the inorder traversals are: ");
inorder(tree);
printf("\n");
break;

case 3 :
printf("\n elements in the preorder 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.");
printf("");
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);
}
}
```