EX 8 : PERFORMING TREE TRAVERSAL TECHNIQUES

Tree traversal

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
struct node *left;
int element;
struct node *right;
};
typedef struct node Node;
Node *Insert(Node *Tree, int e);
void Inorder(Node *Tree);
void Preorder(Node *Tree);
void Postorder(Node *Tree);
int main()
{
Node *Tree = NULL;
int n, i, e, ch;
printf("Enter number of nodes in the tree : ");
scanf("%d", &n);
printf("Enter the elements :\n");
for (i = 1; i <= n; i++)
 {
 scanf("%d", &e);
 Tree = Insert(Tree, e);
 }
do
 {
 printf("1. Inorder \n2. Preorder \n3. Postorder \n4. Exit\n");
 printf("Enter your choice : ");
 scanf("%d", &ch);
 switch (ch)
{
 case 1:
 Inorder(Tree);
 printf("\n");
 break;
 case 2:
 Preorder(Tree);
```

```
 printf("\n");
 break;
 case 3:
 Postorder(Tree);
 printf("\n");
 break;
 }
 } while (ch <= 3);
return 0;
}
Node *Insert(Node *Tree, int e)
{
Node *NewNode = malloc(sizeof(Node));
if (Tree == NULL)
 {
 NewNode->element = e;
 NewNode->left = NULL;
 NewNode->right = NULL;
 Tree = NewNode;
 }
else if (e < Tree->element)
 {
 Tree->left = Insert(Tree->left, e);
 }
else if (e > Tree->element)
 {
 Tree->right = Insert(Tree->right, e);
 }
return Tree;
}
void Inorder(Node *Tree)
{
if (Tree != NULL)
 {
 Inorder(Tree->left);
 printf("%d\t", Tree->element);
 Inorder(Tree->right);
 }
}
void Preorder(Node *Tree)
{
if (Tree != NULL)
 {
 printf("%d\t", Tree->element);
```

```
 Preorder(Tree->left);
 Preorder(Tree->right);
 }
}
void Postorder(Node *Tree)
{
if (Tree != NULL)
 {
 Postorder(Tree->left);
 Postorder(Tree->right);
 printf("%d\t", Tree->element);
 }
}
```

OUTPUT: