**Tree Traversal – Inorder, Preorder, Postorder**

printf("%d ", r->data);

} }

int main()

root = NULL;

int number, value;

printf("Enter the number of elements to be inserted: ");

scanf("%d", &number);

for (int i = 0; i < number; i++) {

printf("Element %d is ", i + 1);

scanf("%d", &value);

root = insert(root, value); }

printf("\nINRODER TARVERSAL\n");

inOrder(root);

printf("\n");

printf("\nPREORDER TRAVERSAL \n");

preOrder(root);

printf("\n");

printf("\nPOSTORDRER TRAVERSAL \n");

postOrder(root);

printf("\n");

return 0;

}

#include <stdio.h>

#include <stdlib.h>

struct node{

int data;

struct node \*left;

struct node \*right;

};

struct node \*root;

struct node \*insert(struct node \*r, int datatonode)

{

if (r == NULL) {

r = (struct node \*)malloc(sizeof(struct node));

r->data = datatonode;

r->left = NULL;

r->right = NULL;

}

else if (datatonode < r->data)

r->left = insert(r->left, datatonode);

else

r->right = insert(r->right, datatonode);

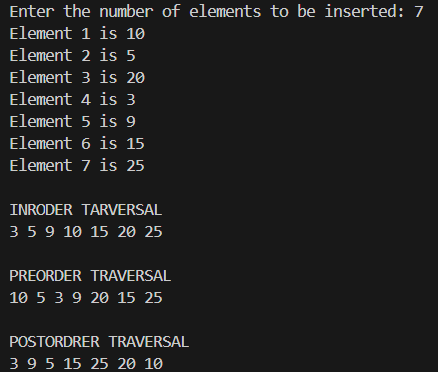
return r;

}

void inOrder(struct node \*r) {

if (r != NULL) {

inOrder(r->left);



printf("%d ", r->data);

inOrder(r->right);

} }

void preOrder(struct node \*r) {

if (r != NULL) {

printf("%d ", r->data);

preOrder(r->left);

preOrder(r->right);

} }

void postOrder(struct node \*r) {

if (r != NULL)

{

postOrder(r->left);

postOrder(r->right);