14.**TREE TRAVERSALS :-**

Code:-

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
  int item;
  struct node* left;
  struct node* right;
};
void inorderTraversal(struct node* root)
 {
  if (root == NULL) return;
  inorderTraversal(root->left);
 printf("%d ->", root->item);
  inorderTraversal(root->right);
}
void preorderTraversal(struct node* root)
 {
  if (root == NULL) return;
  printf("%d ->", root->item);
preorderTraversal(root->left);
  preorderTraversal(root->right);
}
void postorderTraversal(struct node* root)
 {
  if (root == NULL) return;
  postorderTraversal(root->left);
 postorderTraversal(root->right);
  printf("%d ->", root->item);
}
```

```
struct node* createNode(value)
 {
  struct node* newNode = malloc(sizeof(struct node));
  newNode->item = value;
 newNode->left = NULL;
  newNode->right = NULL;

  return newNode;
}
struct node* insertLeft(struct node* root, int value)
 {
  root->left = createNode(value);
  return root->left;
}
struct node* insertRight(struct node* root, int value)
 {
  root->right = createNode(value);
 return root->right;
}
int main()
 {
  struct node* root = createNode(1);
insertLeft(root, 12);
 insertRight(root, 9);
insertLeft(root->left, 5);
 insertRight(root->left, 6);
printf("Inorder traversal \n");
  inorderTraversal(root);
printf("\nPreorder traversal \n");
  preorderTraversal(root);
printf("\nPostorder traversal \n");
```

```
  postorderTraversal(root);
}
```

OUTPUT:-