|  |
| --- |
| **Part A**  **Name:- Aryan Srivastava**  **Roll No:- A073**  **Subject:- Data Structures and Algorithms**  **Program: B Tech/MBA Tech CE 2nd Year** |
| **Aim:**  To study and implement Binary Tree traversal algorithms. |
| **Prerequisite:** C++ Programming |
| **Outcome:** Representation of Binary trees in memory and implementation of Binary Tree traversal algorithms |
| * **Theory:** * Binary tree * Tree with 0, 1 or atmost 2 children per node. * A node that has zero children is called a leaf node or a terminal node. * Every node contains a data element, a left pointer which points to the left child, and a right pointer which points to the right child. * The root element is pointed by a 'root' pointer. If root = NULL, then it means the tree is empty     **Binary tree- Node representation**   * struct node { * struct node \*left; * int data; * struct node \*right; * };     **TRAVERSING A BINARY TREE**   * Traversing a binary tree is the process of visiting each node in the tree exactly once in a systematic way. * Unlike linear data structures in which the elements are traversed sequentially, tree is a nonlinear data structure in which the elements can be traversed in many different ways. * There are different algorithms for tree traversals: * Inorder traversal * Postorder traversal * Preorder traversal   **Preorder traversal:**  **struct node**  **{    int data;**  **struct node \*left;**  **struct node \*right;**  **};**  **struct Node \*root = NULL;**  **void preorder(struct node \*root)**  **{    if (root != NULL)**  **{**  **cout<<root->data<<" ";**  **preorder(root->left);**  **preorder(root->right);**  **}**  **}**  **// Inorder traversal**  **void inorder(struct node \*root)**  **{    if (root != NULL)**  **{**  **inorder(root->left);**  **cout<<root->data<<" ";**  **inorder(root->right);**  **}**  **}**  **// Postorder traversal**  **void postorder(struct node \*root)**  **{    if (root != NULL)**  **{**  **postorder(root->left);**  **postorder(root->right);**  **cout<<root->data<<" ";**  **}**  **}**  **//Insert**  **struct node \*insert(struct node \*root, int val)**  **{**  **if(root == NULL)**  **return getNewNode(val);**  **if(root->key < val)**  **root->right = insert(root->right,val);**  **else if(root->key > val)**  **root->left = insert(root->left,val);**  **return root;**    **}**  **struct node \*getNewNode(int val)**  **{**  **struct node \*newNode = new node;**  **newNode->key = val;**  **newNode->left = NULL;**  **newNode->right = NULL;**  **return newNode;**  **}**  **TASK 1:**  Write a C/C++ program to implement binary tree and perfrom the following operations:   1. Insert 2. Inorder traversal 3. Preorder traversal 4. Postorder traversal |
| **Procedure:**   1. Open CodeBlock editor or visual studio editor and write the code in C++. 2. Complile and run the code |
| **Instructions:**   1. Copy code & paste in code section and output of Part B. |
|  |

|  |
| --- |
| **Part B** |
| Code:  class Node {      int value;      Node left, right;      Node(int item) {          value = item;          left = right = null;      }  }  class BinaryTree {      Node root;      BinaryTree() {          root = null;      }      void insert(int value) {          root = insertingNode(root, value);      }      Node insertingNode(Node root, int value) {          if (root == null) {              root = new Node(value);              return root;          }          if (value < root.value) {              root.left = insertingNode(root.left, value);          } else if (value > root.value) {              root.right = insertingNode(root.right, value);          }          return root;      }      void preorder() {          preordering(root);          System.out.println();      }      void preordering(Node root) {          if (root != null) {              System.out.print(root.value + " ");              preordering(root.left);              preordering(root.right);          }      }      void postorder() {          postordering(root);          System.out.println();      }      void postordering(Node root) {          if (root != null) {              postordering(root.left);              postordering(root.right);              System.out.print(root.value + " ");          }      }      public static void main(String[] args) {          BinaryTree bst = new BinaryTree();          bst.insert(50);          bst.insert(40);          bst.insert(30);          bst.insert(20);          bst.insert(70);          bst.insert(60);          bst.insert(80);          System.out.print("preorder: ");          bst.preorder();          System.out.print("postorder: ");          bst.postorder();      }  } |
| **Output:** |
| **Observation & Learning:**  Successgully studied and implemented Binary Tree traversal algorithms. |
| Curiosity questions:   1. What is the time complexity of PUSH operation in stack using a Singly Linked list? a) O (n) b) O (n2) c) O (nlogn) d) O (1) 2. 1. What is the time complexity of POP operation in stack using a Singly Linked list? a) O (n) b) O (n2) c) O (nlogn) d) O (1) |
| **Conclusion:**  We successfully implemented stack its operations usng linked list and using C++ program. |