# **Bansilal Ramnath Agarwal Charitable Trust’s**

**Vishwakarma Institute of Technology, Pune-37**

*(An Autonomous Institute of Savitribai Phule Pune University)*

|  |  |
| --- | --- |
|  |  |
|  |  |

**Department of Artificial Intelligence and Data Science**

|  |  |
| --- | --- |
| **Division** | AI-A |
| **Batch** | 1 |
| **GR-no** | 12211542 |
| **Roll No.** | 07 |
| **Name** | Anish Naphade |
| **Subject** | ADS |

**ASSIGNMENT 6 –**

**Implement BST with following options**

**1. Insert**

**2. Delete**

**3. Traversal Inorder, Preorder,postorder with and without recursion**

**CODE**

**#include<stdio.h>**

**#include<stdlib.h>**

**struct TreeNode {**

**int data;**

**struct TreeNode\* left;**

**struct TreeNode\* right;**

**};**

**struct StackNode {**

**struct TreeNode\* treeNode;**

**struct StackNode\* next;**

**};**

**struct TreeNode\* createNode(int value) {**

**struct TreeNode\* newNode = (struct TreeNode\*)malloc(sizeof(struct TreeNode));**

**newNode->data = value;**

**newNode->left = NULL;**

**newNode->right = NULL;**

**return newNode;**

**}**

**struct StackNode\* createStackNode(struct TreeNode\* treeNode) {**

**struct StackNode\* stackNode = (struct StackNode\*)malloc(sizeof(struct StackNode));**

**stackNode->treeNode = treeNode;**

**stackNode->next = NULL;**

**return stackNode;**

**}**

**void push(struct StackNode\*\* top\_ref, struct TreeNode\* treeNode) {**

**struct StackNode\* stackNode = createStackNode(treeNode);**

**stackNode->next = \*top\_ref;**

**\*top\_ref = stackNode;**

**}**

**struct TreeNode\* pop(struct StackNode\*\* top\_ref) {**

**if (\*top\_ref == NULL)**

**return NULL;**

**struct TreeNode\* result = (\*top\_ref)->treeNode;**

**struct StackNode\* temp = \*top\_ref;**

**\*top\_ref = (\*top\_ref)->next;**

**free(temp);**

**return result;**

**}**

**int isEmpty(struct StackNode\* top) {**

**return top == NULL;**

**}**

**struct TreeNode\* insert(struct TreeNode\* root, int value) {**

**if (root == NULL) {**

**return createNode(value);**

**}**

**if (value < root->data) {**

**root->left = insert(root->left, value);**

**} else if (value > root->data) {**

**root->right = insert(root->right, value);**

**}**

**return root;**

**}**

**void printInOrder(struct TreeNode\* node) {**

**if (node != NULL) {**

**printInOrder(node->left);**

**printf("%d ", node->data);**

**printInOrder(node->right);**

**}**

**}**

**void printPreOrder(struct TreeNode\* node) {**

**if(node != NULL) {**

**printf("%d ",node->data);**

**printPreOrder(node->left);**

**printPreOrder(node->right);**

**}**

**}**

**void printPostOrder(struct TreeNode\* node) {**

**if(node != NULL) {**

**printPostOrder(node->left);**

**printPostOrder(node->right);**

**printf("%d ",node->data);**

**}**

**}**

**struct TreeNode\* findMin(struct TreeNode\* node) {**

**while (node->left != NULL) {**

**node = node->left;**

**}**

**return node;**

**}**

**void inorderTraversal(struct TreeNode\* root) {**

**if (root == NULL)**

**return;**

**struct StackNode\* stack = NULL;**

**struct TreeNode\* current = root;**

**while (current != NULL || !isEmpty(stack)) {**

**while (current != NULL) {**

**push(&stack, current);**

**current = current->left;**

**}**

**current = pop(&stack);**

**printf("%d ", current->data);**

**current = current->right;**

**}**

**}**

**void preorderTraversal(struct TreeNode\* root) {**

**if (root == NULL)**

**return;**

**struct StackNode\* stack = NULL;**

**push(&stack, root);**

**while (!isEmpty(stack)) {**

**struct TreeNode\* current = pop(&stack);**

**printf("%d ", current->data);**

**if (current->right != NULL)**

**push(&stack, current->right);**

**if (current->left != NULL)**

**push(&stack, current->left);**

**}**

**}**

**void postorderTraversal(struct TreeNode\* root) {**

**if (root == NULL)**

**return;**

**struct StackNode\* stack1 = NULL;**

**struct StackNode\* stack2 = NULL;**

**push(&stack1, root);**

**while (!isEmpty(stack1)) {**

**struct TreeNode\* current = pop(&stack1);**

**push(&stack2, current);**

**if (current->left != NULL)**

**push(&stack1, current->left);**

**if (current->right != NULL)**

**push(&stack1, current->right);**

**}**

**while (!isEmpty(stack2)) {**

**struct TreeNode\* current = pop(&stack2);**

**printf("%d ", current->data);**

**}**

**}**

**struct TreeNode\* deleteNode(struct TreeNode\* root, int value) {**

**if (root == NULL) {**

**return root;**

**}**

**if (value < root->data) {**

**root->left = deleteNode(root->left, value);**

**} else if (value > root->data) {**

**root->right = deleteNode(root->right, value);**

**} else {**

**if (root->left == NULL) {**

**struct TreeNode\* temp = root->right;**

**free(root);**

**return temp;**

**} else if (root->right == NULL) {**

**struct TreeNode\* temp = root->left;**

**free(root);**

**return temp;**

**}**

**struct TreeNode\* temp = findMin(root->right);**

**root->data = temp->data;**

**root->right = deleteNode(root->right, temp->data);**

**}**

**return root;**

**}**

**int main() {**

**struct TreeNode\* root = NULL;**

**int choice, data;**

**while (1) {**

**printf("1. Insert element\n2. InOrder Traversal\n3. Pre-Oreder Traversal \n4. Post-order Traversal \n5. InOrder without Recursion \n6. Pre-Order without Recursion \n7. Post-Order without Recursion\n8. Delete element\n9. Exit\nEnter Choice: ");**

**scanf("%d", &choice);**

**switch (choice) {**

**case 1: {**

**printf("Enter data to insert: ");**

**scanf("%d", &data);**

**root = insert(root, data);**

**break;**

**}**

**case 2: {**

**printf("In-order traversal: ");**

**printInOrder(root);**

**printf("\n");**

**break;**

**}**

**case 3: {**

**printf("Pre-order traversal: ");**

**printPostOrder(root);**

**printf("\n");**

**break;**

**}**

**case 4: {**

**printf("Post-order traversal: ");**

**printPostOrder(root);**

**printf("\n");**

**break;**

**}**

**case 5: {**

**printf("InOrder without Recursion: ");**

**inorderTraversal(root);**

**printf("\n");**

**break;**

**}**

**case 6: {**

**printf("Pre-Order without Recursion: ");**

**preorderTraversal(root);**

**printf("\n");**

**break;**

**}**

**case 7: {**

**printf("Post-Order without Recursion: ");**

**postorderTraversal(root);**

**printf("\n");**

**break;**

**}**

**case 8: {**

**printf("Enter data to delete: ");**

**scanf("%d", &data);**

**root = deleteNode(root, data);**

**break;**

**}**

**case 9: {**

**printf("Exiting program.\n");**

**return 0;**

**}**

**default: {**

**printf("Wrong input\n");**

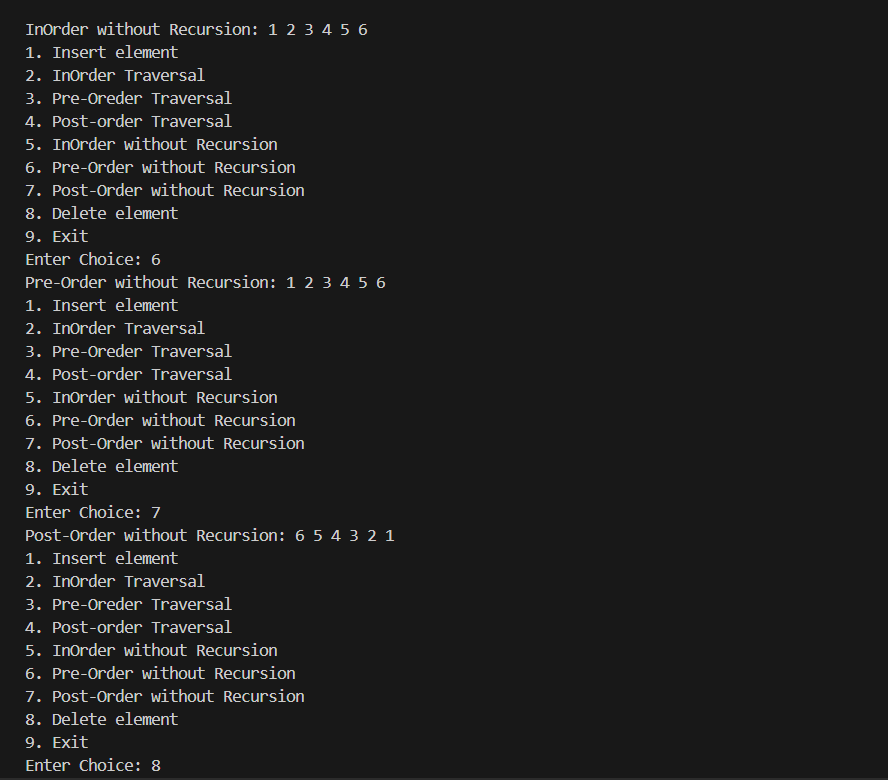
**}**

**}**

**}**

**return 0;**

**}**

****