**EX.NO: 6**

**IMPLEMENTATION OF BINARY TREES AND OPERATIONS OF BINARY TREES**

**DATE:**

**AIM:**

**To write a c program to implement binary tree and its operations.**

**ALGORITHM:**

**1.Start the program.**

**2.Declare the struct node.**

**3.Declare the function insertion,deletion and display.**

**4.Insertion function is use to insert the value to the tree.**

**5.Delete function is use to delete the value from the tree.**

**6.Display function is use to print the tree.**

**7.Stop**

**PROGRAM**

**#include<stdio.h>**

**#include<stdlib.h>**

**struct node**

**{**

**int value;**

**struct node \*left\_child, \*right\_child;**

**};**

**struct node \*new\_node(int value)**

**{**

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

**tmp->value = value;**

**tmp->left\_child = tmp->right\_child = NULL;**

**return tmp;**

**}**

**void delete\_tree(struct node \*root\_node)**

**{**

**if (root\_node)**

**{**

**delete\_tree(root\_node->left\_child);**

**delete\_tree(root\_node->right\_child);**

**free(root\_node);**

**}**

**}**

**void print(struct node \*root\_node)**

**{**

**if (root\_node != NULL)**

**{**

**print(root\_node->left\_child);**

**printf("%d -->", root\_node->value);**

**print(root\_node->right\_child);**

**}**

**}**

**struct node\* insertNode(struct node\* node, int value) // inserting nodes!**

**{**

**if (node == NULL)**

**return new\_node(value);**

**if (value < node->value)**

**node->left\_child = insertNode(node->left\_child, value);**

**else if (value > node->value)**

**node->right\_child = insertNode(node->right\_child, value);**

**return node;**

**}**

**int main()**

**{**

**struct node \*root = NULL;**

**int data;**

**char ch;**

**printf("\n 1.Create Tree \n 2.Display \n 3.Delete \n 4.Exit");**

**do**

**{**

**int choice;**

**printf("\n Enter ur choice");**

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

**switch (choice)**

**{**

**case 1 :**

**printf("\nEnter the value to be inserted\n");**

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

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

**break;**

**case 2 :**

**printf("\n Construct Binary Tree::\n");**

**print(root);**

**break;**

**case 3 :**

**delete\_tree(root);**

**printf("\n All data are removed");**

**break;**

**default :**

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

**break;**

**}**

**printf("\nDo you want to continue (Type y or n)\n");**

**scanf(" %c",&ch);**

**} while (ch == 'Y'|| ch == 'y');**

**return 0;**

**}**

**OUTPUT**

**1.Create Tree**

**2.Display**

**3.Delete**

**4.Exit**

**Enter ur choice1**

**Enter the value to be inserted**

**5**

**Do you want to continue (Type y or n)**

**Y**

**Enter ur choice1**

**Enter the value to be inserted**

**2**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice1**

**Enter the value to be inserted**

**1**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice1**

**Enter the value to be inserted**

**3**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice1**

**Enter the value to be inserted**

**6**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice1**

**Enter the value to be inserted**

**7**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice2**

**Construct Binary Tree::**

**1 -->2 -->3 -->5 -->6 -->7 -->**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice3**

**All data are removed**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice4**

**Wrong Entry**

**RESULT:**

**Thus the above c program of Binary Tree and its operation implementation is successfully.**

**EX.NO: 7**

**IMPLEMENTATION OF BINARY SEARCH TREES**

**DATE:**

**AIM:**

**To write a c program to implement binary tree and its operations.**

**ALGORITHM:**

**1.Start the program.**

**2.Declare the struct node.**

**3.Declare the function insertion,deletion and search.**

**4.Insertion function is use to insert the value to the tree.**

**5.Search function is use to found the value or not.**

**6.Delete function is use to delete the value from the tree.**

**7.Display function is use to print the tree.**

**8.Stop**

**PROGRAM**

**#include<stdio.h>**

**#include<stdlib.h>**

**struct node**

**{**

**int value;**

**struct node \*left\_child, \*right\_child;**

**};**

**struct node \*new\_node(int value)**

**{**

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

**temp->value = value;**

**temp->left\_child = temp->right\_child = NULL;**

**return temp;**

**}**

**struct node\* search(struct node \*root\_node, int x)**

**{**

**if(root\_node==NULL || root\_node->value==x)**

**return root\_node;**

**else if(x>root\_node->value)**

**return search(root\_node->right\_child,x);**

**else**

**return search(root\_node->left\_child,x);**

**}**

**//function to find the minimum value in a node**

**struct node\* find\_minimum(struct node \*root\_node)**

**{**

**if(root\_node == NULL)**

**return NULL;**

**else if(root\_node->left\_child != NULL) // node with minimum value will have no left child**

**return find\_minimum(root\_node->left\_child); // left most element will be minimum**

**return root\_node;**

**}**

**void print(struct node \*root\_node)**

**{**

**if (root\_node != NULL)**

**{**

**print(root\_node->left\_child);**

**printf("%d -->", root\_node->value);**

**print(root\_node->right\_child);**

**}**

**}**

**struct node\* insertNode(struct node\* node, int value) // inserting nodes!**

**{**

**if (node == NULL)**

**return new\_node(value);**

**if (value < node->value)**

**node->left\_child = insertNode(node->left\_child, value);**

**else if (value > node->value)**

**node->right\_child = insertNode(node->right\_child, value);**

**return node;**

**}**

**struct node\* delete(struct node \*root\_node, int x)**

**{**

**if(root\_node==NULL)**

**return NULL;**

**if (x>root\_node->value)**

**root\_node->right\_child = delete(root\_node->right\_child, x);**

**else if(x<root\_node->value)**

**root\_node->left\_child = delete(root\_node->left\_child, x);**

**else**

**{**

**//No Children**

**if(root\_node->left\_child==NULL && root\_node->right\_child==NULL)**

**{**

**free(root\_node);**

**return NULL;**

**}**

**//One Child**

**else if(root\_node->left\_child==NULL || root\_node->right\_child==NULL)**

**{**

**struct node \*temp;**

**if(root\_node->left\_child==NULL)**

**temp = root\_node->right\_child;**

**else**

**temp = root\_node->left\_child;**

**free(root\_node);**

**return temp;**

**}**

**//Two Children**

**else**

**{**

**struct node \*temp = find\_minimum(root\_node->right\_child);**

**root\_node->value = temp->value;**

**root\_node->right\_child = delete(root\_node->right\_child, temp->value);**

**}**

**}**

**return root\_node;**

**}**

**int main()**

**{**

**struct node \*root = NULL;**

**int data;**

**char ch;**

**printf("\n 1.Create Tree \n 2.Construct Tree \n 3.Delete \n 4.Search \n 5.Exit");**

**do**

**{**

**int choice;**

**printf("\n Enter ur choice");**

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

**switch (choice)**

**{**

**case 1 :**

**printf("\nEnter the value to be inserted\n");**

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

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

**break;**

**case 2 :**

**printf("\n Construct Tree::\n");**

**print(root);**

**break;**

**case 3 :**

**printf("\nEnter the value to be deleted\n");**

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

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

**print(root);**

**break;**

**case 4:**

**printf("\n Enter the value to be searched");**

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

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

**print(root);**

**break;**

**default :**

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

**break;**

**}**

**printf("\nDo you want to continue (Type y or n)\n");**

**scanf(" %c",&ch);**

**} while (ch == 'Y'|| ch == 'y');**

**return 0;**

**}**

**OUTPUT**

**1.Create Tree**

**2.Construct Tree**

**3.Delete**

**4.Search**

**5.Exit**

**Enter ur choice1**

**Enter the value to be inserted**

**5**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice1**

**Enter the value to be inserted**

**2**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice1**

**Enter the value to be inserted**

**1**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice1**

**Enter the value to be inserted**

**3**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice1**

**Enter the value to be inserted**

**6**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice1**

**Enter the value to be inserted**

**7**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice2**

**Construct Tree::**

**1 -->2 -->3 -->5 -->6 -->7 -->**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice3**

**Enter the value to be deleted**

**3**

**1 -->2 -->5 -->6 -->7 -->**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice4**

**Enter the value to be searched6**

**6 -->7 -->**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice5**

**Wrong Entry**

**RESULT:**

**Thus the above c program of Binary Search Tree implementation is successfully.**

**EX.NO: 8**

**IMPLEMENTATION OF TREE TRAVERSAL**

**DATE:**

**AIM:**

**To write a c program to implement Binary Tree Traversal.**

**ALGORITHM:**

**1.Start the program.**

**2.Declare the struct node.**

**3.Declare the function in order,pre order and post order.**

**4.In order function perform (LDR)**

**5.Pre order function perform (DLR)**

**6. Post order function perform (LRD)**

**7.Display function is use to print the tree.**

**8.Stop**

**#include<stdio.h>**

**#include<stdlib.h>**

**struct node**

**{**

**int value;**

**struct node \*left\_child, \*right\_child;**

**};**

**struct node \*new\_node(int value)**

**{**

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

**tmp->value = value;**

**tmp->left\_child = tmp->right\_child = NULL;**

**return tmp;**

**}**

**struct node\* insertNode(struct node\* node, int value)**

**{**

**if (node == NULL)**

**return new\_node(value);**

**if (value < node->value)**

**node->left\_child = insertNode(node->left\_child, value);**

**else if (value > node->value)**

**node->right\_child = insertNode(node->right\_child, value);**

**return node;**

**}**

**void preorder(struct node \* root\_node)**

**{**

**if (root\_node)**

**{**

**printf("%d-->",root\_node->value);**

**preorder(root\_node->left\_child);**

**preorder(root\_node->right\_child);**

**}**

**}**

**void inorder(struct node \* root\_node)**

**{**

**if (root\_node)**

**{**

**inorder(root\_node->left\_child);**

**printf("%d-->",root\_node->value);**

**inorder(root\_node->right\_child);**

**}**

**}**

**void postorder(struct node \* root\_node)**

**{**

**if (root\_node)**

**{**

**postorder(root\_node->left\_child);**

**postorder(root\_node->right\_child);**

**printf("%d-->",root\_node->value);**

**}**

**}**

**int main()**

**{**

**struct node \*root = NULL;**

**int data;**

**char ch;**

**printf("\n 1.Create Tree \n 2.Inorder Traversal \n 3.Preorder Traversal \n**

**4.Postorder Traversal \n 5.Exit");**

**do**

**{**

**int choice;**

**printf("\n Enter ur choice");**

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

**switch (choice)**

**{**

**case 1 :**

**printf("\nEnter the value to be inserted\n");**

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

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

**break;**

**case 2 :**

**printf("\n Inorder Traversal::\n");**

**inorder(root);**

**break;**

**case 3 :**

**printf("\n Preorder Traversal::\n");**

**preorder(root);**

**break;**

**case 4 :**

**printf("\n Postorder Traversal::\n");**

**postorder(root);**

**break;**

**default :**

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

**break;**

**}**

**printf("\nDo you want to continue (Type y or n)\n");**

**scanf(" %c",&ch);**

**} while (ch == 'Y'|| ch == 'y');**

**return 0;**

**}**

**OUTPUT**

**1.Create Tree**

**2.Inorder Traversal**

**3.Preorder Traversal**

**4.Postorder Traversal**

**5.Exit**

**Enter ur choice1**

**Enter the value to be inserted**

**5**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice1**

**Enter the value to be inserted**

**2**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice1**

**Enter the value to be inserted**

**1**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice1**

**Enter the value to be inserted**

**3**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice1**

**Enter the value to be inserted**

**6**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice1**

**Enter the value to be inserted**

**7**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice2**

**Inorder Traversal::**

**1-->2-->3-->5-->6-->7-->**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice3**

**Preorder Traversal::**

**5-->2-->1-->3-->6-->7-->**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice4**

**Postorder Traversal::**

**1-->3-->2-->7-->6-->5-->**

**Do you want to continue (Type y or n)**

**y**

**Enter ur choice5**

**Wrong Entry\**