**Program 8:** **Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers.**

**a. Create a BST of N Integers**

**b. Traverse the BST in Inorder, Preorder and Post Order**

**c. Search the BST for a given element (KEY) and report the appropriate message d. Exit**

**#include<stdio.h>**

**#include<malloc.h>**

**struct node**

**{**

**int info;**

**struct node \*left;**

**struct node \*right;**

**};**

**typedef struct node \* NODE;**

**//-----------------------------------------------------------------------------------------------------------------**

**NODE create(NODE root)**

**{**

**NODE newnode,pres,prev;**

**newnode = ( NODE ) malloc(sizeof(struct node));**

**printf(" Enter the info to newnode \n");**

**scanf("%d", &newnode->info);**

**newnode->left = newnode->right = NULL;**

**if(root == NULL)**

**{**

**root = newnode;**

**return root;**

**}**

**pres = root;**

**while( pres != NULL)**

**{**

**if(newnode->info == pres->info)**

**{**

**printf(" Duplicate Identifiers are not allowed \n");**

**return root;**

**}**

**prev = pres;**

**if(newnode->info < pres->info)**

**pres = pres->left;**

**else**

**pres = pres->right;**

**}**

**if(newnode->info < prev->info)**

**prev->left = newnode;**

**else**

**prev->right = newnode;**

**return root;**

**}**

**//--------------------------------------------------------------------------**

**void search(NODE root, int key)**

**{**

**NODE pres,flag = 0;**

**if(root == NULL)**

**printf(" The Tree is empty, search cannot be done \ ");**

**else**

**void preorder(NODE root)**

**{**

**if( root != NULL)**

**{**

**printf("%d\t", root->info);**

**preorder(root->left);**

**preorder(root->right);**

**}**

**}**

**{**

**pres = root;**

**while( pres != NULL)**

**{**

**if(key == pres->info)**

**{**

**flag = 1;**

**printf(" Key is found \n");**

**return;**

**void inorder(NODE root)**

**{**

**if( root != NULL)**

**{**

**inorder(root->left);**

**printf("%d\t", root->info);**

**inorder(root->right);**

**}**

**}**

**}**

**if(key < pres->info)**

**pres = pres->left ;**

**else**

**pres = pres->right;**

**}**

**}**

**if(flag == 0)**

**printf(" Key not found \n");**

**}**

**void postorder(NODE root)**

**{**

**if( root != NULL)**

**{**

**postorder(root->left);**

**postorder(root->right);**

**printf("%d\t", root->info);**

**}**

**}**

**void main()**

**{**

**NODE root = NULL;**

**int ch,key;**

**for(;;)**

**{**

**printf("\n Enter 1: Create 2:Preorder 3:Inorder 4:Postorder \n");**

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

**switch(ch)**

**{**

**case 1: root = create(root);**

**break;**

**case 2: if( root == NULL)**

**printf(" The Tree is Empty \n");**

**else**

**{**

**printf("\n The Preorder Traversal is \n");**

**preorder(root);**

**}**

**break;**

**case 3: if( root == NULL)**

**printf(" The Tree is Empty \n");**

**else**

**{**

**printf("\n The Inorder Traversal is \n");**

**inorder(root);**

**}**

**break;**

**case 4: if( root == NULL)**

**printf(" The Tree is Empty \n");**

**else**

**{**

**printf("\n The Postorder Traversal is \n");**

**postorder(root);**

**}**

**break;**

**case 5: printf(" Enter the key to be searched \n");**

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

**search(root,key);**

**break;**

**default: exit();**

**}**

**}**

**}**