Ex 9: Binary Search Tree

REGISTER.NO:-231801155 **NAME:-SARANYA V**

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
struct BinaryTreeNode {
   int key;
   struct BinaryTreeNode *left, *right;
};

struct BinaryTreeNode* newNodeCreate(int value)
{
   struct BinaryTreeNode* temp
      = (struct BinaryTreeNode*)malloc(
         sizeof(struct BinaryTreeNode));
   temp->key = value;
   temp->left = temp->right = NULL;
   return temp;
}
struct BinaryTreeNode*
searchNode(struct BinaryTreeNode* root, int target)
{
   if (root == NULL || root->key == target) {
      return root;
   }
```

```
    if (root->key < target) {
        return searchNode(root->right, target);
    }
    return searchNode(root->left, target);
}
struct BinaryTreeNode*
insertNode(struct BinaryTreeNode* node, int value)
{
    if (node == NULL) {
        return newNodeCreate(value);
    }
    if (value < node->key) {
        node->left = insertNode(node->left, value);
    }
    else if (value > node->key) {
        node->right = insertNode(node->right, value);
    }
    return node;
}
void postOrder(struct BinaryTreeNode* root)
{
    if (root != NULL) {
        postOrder(root->left);
        postOrder(root->right);
        printf(" %d ", root->key);
    }
}
```

```
void inOrder(struct BinaryTreeNode* root)
{
   if (root != NULL) {
      inOrder(root->left);
      printf(" %d ", root->key);
      inOrder(root->right);
   }
}
void preOrder(struct BinaryTreeNode* root)
{
   if (root != NULL) {
      printf(" %d ", root->key);
      preOrder(root->left);
      preOrder(root->right);
   }
}
struct BinaryTreeNode* findMin(struct BinaryTreeNode* root)
{
   if (root == NULL) {
      return NULL;
   }
   else if (root->left != NULL) {
      return findMin(root->left);
   }
   return root;
}
struct BinaryTreeNode* delete (struct BinaryTreeNode* root,
```

```
                        int x)
{
    if (root == NULL)
        return NULL;

    if (x > root->key) {
        root->right = delete (root->right, x);
    }
    else if (x < root->key) {
        root->left = delete (root->left, x);
    }
    else {
        if (root->left == NULL && root->right == NULL) {
            free(root);
            return NULL;
        }
        else if (root->left == NULL
                || root->right == NULL) {
            struct BinaryTreeNode* temp;
            if (root->left == NULL) {
                temp = root->right;
            }
            else {
                temp = root->left;
            }
            free(root);
            return temp;
```

```
        }
        else {
            struct BinaryTreeNode* temp
                = findMin(root->right);
            root->key = temp->key;
            root->right = delete (root->right, temp->key);
        }
    }
    return root;
}

int main()
{
    struct BinaryTreeNode* root = NULL;
    root = insertNode(root, 50);
    insertNode(root, 30);
    insertNode(root, 20);
    insertNode(root, 40);
    insertNode(root, 70);
    insertNode(root, 60);
    insertNode(root, 80);
    if (searchNode(root, 60) != NULL) {
        printf("60 found");
    }
    else {
        printf("60 not found");
    }
```

```
    printf("\n");
    postOrder(root);
    printf("\n");
    preOrder(root);
    printf("\n");
    inOrder(root);
    printf("\n");
    struct BinaryTreeNode* temp = delete (root, 70);
    printf("After Delete: \n");
    inOrder(root);


    return 0;
}
```

OUTPUT: