SRN : PES1PG22CA239
NAME : VINAYAK SUBRAY HEGDE

CODE :

```
#include <stdio.h>
#include <stdlib.h>

typedef struct tree
{
struct Node *root;
} tree;

typedef struct Node
{
struct Node *left, *right;
int data;
} Node;

Node *getNode(int data)
{
Node *newNode = (Node *)malloc(sizeof(Node));
newNode->data = data;
newNode->left = newNode->right = NULL;
return newNode;
}

Node *buildBST(Node *root, int val)
{
if (root == NULL)
{
root = getNode(val);
return root;
}
if (root->data > val)
{
root->left = buildBST(root->left, val);
}
else
{
root->right = buildBST(root->right, val);
}

return root;
}

void inOrder(Node *root)
{
if (root == NULL)
return;
```

SRN : PES1PG22CA239
NAME : VINAYAK SUBRAY HEGDE

```
inOrder(root->left);
printf("%d ", root->data);
inOrder(root->right);
}

void preOrder(Node *root)
{
if (root == NULL)
return;
printf("%d ", root->data);
preOrder(root->left);
preOrder(root->right);
}

void postOrder(Node *root)
{
if (root == NULL)
return;
postOrder(root->left);
postOrder(root->right);
printf("%d ", root->data);
}

Node* deleteByValue(Node* root, int val)
{
if (root == NULL)
{
return root;
}
if (val < root->data)
{
root->left = deleteByValue(root->left, val);
}
else if (val > root->data)
{
root->right = deleteByValue(root->right, val);
}
else
{
// Case 1: Node with only one child or no child
if (root->left == NULL)
{
Node* temp = root->right;
free(root);
return temp;
}
else if (root->right == NULL)
{
Node* temp = root->left;
```

SRN : PES1PG22CA239
NAME : VINAYAK SUBRAY HEGDE

```
free(root);
return temp;
}

// Case 2: Node with two children
Node* temp = root->right;
while (temp && temp->left != NULL)
{
temp = temp->left;
}

root->data = temp->data;
root->right = deleteByValue(root->right, temp->data);
}

return root;
}

Node *search(Node *root, int val)
{
if (root == NULL || root->data == val)
{
return root;
}
if (root->data > val)
{
return search(root->left, val);
}
else
{
return search(root->right, val);
}
}

int main()
{
tree tree;
int choice, val, deletevalue;
int arr[] = {40, 23, 4, 7, 23, 12, 5, 1, 354, 45, 100, 389};
int length = sizeof(arr) / sizeof(arr[0]);
tree.root = NULL;

for (int i = 0; i < length; i++)
{
tree.root = buildBST(tree.root, arr[i]);
}

do {
printf("\nBinary Search Tree Operations\n");
```

SRN : PES1PG22CA239
NAME : VINAYAK SUBRAY HEGDE

```
printf("----------------------------\n");
printf("1. Insert a node\n");
printf("2. Delete a node\n");
printf("3. Search for a node\n");
printf("4. Traverse the tree\n");
printf("5. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);

switch (choice) {
case 1:
printf("Enter the value to insert: ");
scanf("%d", &val);
tree.root = buildBST(tree.root, val);
break;
case 2:
printf("Enter the value to delete: ");
scanf("%d", &deletevalue);
tree.root = deleteByValue(tree.root, deletevalue);
break;
case 3:
printf("Enter the value to search: ");
scanf("%d", &val);
Node *result = search(tree.root, val);
if (result == NULL) {
printf("Value not found\n");
} else {
printf("Value found: %d\n", result->data);
}
break;
case 4:
printf("\nIn-order traversal: ");
inOrder(tree.root);
printf("\nPre-order traversal: ");
preOrder(tree.root);
printf("\nPost-order traversal: ");
postOrder(tree.root);
printf("\n");
break;
case 5:
printf("Exiting program...\n");
break;
default:
printf("Invalid choice\n");
}
} while (choice != 5);

return 0;
}
```

SRN : PES1PG22CA239
NAME : VINAYAK SUBRAY HEGDE

Output :