ANVITA KUMAR
C-22
Roll No.: 2104097

```
// Write a menu driven code to implement Binary Search Tree

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

struct node {

   int data;

   struct node *left;

   struct node *right;

};

struct node *tree;

void create_tree(struct node *);

struct node *insert(struct node *, int);

struct node *delete (struct node *, int);

struct node *search(struct node *, int);

void preorderTraversal(struct node *);

void inorderTraversal(struct node *);

void postorderTraversal(struct node *);

int totalNodes(struct node *);

int totalLeafNodes(struct node *);

int totalInternalNodes(struct node *);

int Height(struct node *);

int main()

{

   int option, val;

   create_tree(tree);

   do {

      printf("\n***List Of Operations***");

      printf("\n1. Insertion\n2. Deletion\n3. Searching\n4. Pre-order Traversal\n5. In-order
Traversal\n6. Postorder Traversal\n7. Total number of nodes\n8. Total number of leaf nodes\n9.
Total number of internal nodes\n10. Find height of the tree\n11. Exit\n");

      printf("Enter your option : ");
```

ANVITA KUMAR
C-22
Roll No.: 2104097

```
    scanf("%d", &option);
    switch (option) {
    case 1:
      printf("Enter the value to be inserted: ");
      scanf("%d", &val);
      tree = insert(tree, val);
      break;
    case 2:
      printf("Enter the element to be deleted: ");
      scanf("%d", &val);
      tree = delete (tree, val);
      break;
    case 3:
      printf("Enter the element to be searched: ");
      scanf("%d", &val);
      tree = search(tree, val);
      if(tree)
        printf("The value %d is found in the tree",val);
      else
        printf("The value %d not found",val);
      break;
    case 4:
      printf("The elements of the tree are : \n");
      preorderTraversal(tree);
      break;
    case 5:
      printf("The elements of the tree are : \n");
      inorderTraversal(tree);
      break;
    case 6:
      printf("The elements of the tree are : \n");
```

```
            postorderTraversal(tree);
            break;
        case 7:
            printf("Total no. of nodes = %d", totalNodes(tree));
            break;
        case 8:
            printf("Total no. of leaf nodes = %d",
                totalLeafNodes(tree));
            break;
        case 9:
            printf("Total no. of internal nodes = %d",
                totalInternalNodes(tree));
            break;
        case 10:
            printf("The height of the tree = %d", Height(tree));
            break;
        case 11:
            printf("\n\tEXIT POINT!");
            break;
        }
    } while (option != 11);
    return 0;
}
void create_tree(struct node *tree)
{
    tree = NULL;
}
struct node *insert(struct node *tree, int val)
{
    struct node *ptr, *nodeptr, *parentptr;
    ptr = (struct node *)malloc(sizeof(struct node));
```

ANVITA KUMAR
C-22
Roll No.: 2104097

```
    ptr->data = val;
    ptr->left = NULL;
    ptr->right = NULL;
    if (tree == NULL) {
        tree = ptr;
        tree->left = NULL;
        tree->right = NULL;
    }
    else {
        parentptr = NULL;
        nodeptr = tree;
        while (nodeptr != NULL) {
            parentptr = nodeptr;
            if (val < nodeptr->data)
                nodeptr = nodeptr->left;
            else
                nodeptr = nodeptr->right;
        }
        if (val < parentptr->data)
            parentptr->left = ptr;
        else
            parentptr->right = ptr;
    }
    return tree;
}
struct node *delete (struct node *tree, int val)
{
    struct node *cur, *parent, *suc, *psuc, *ptr;
    if (tree->left == NULL) {
        printf("\nThe tree is empty");
        return (tree);
```

ANVITA KUMAR
C-22
Roll No.: 2104097

```
    }
    parent = tree;
    cur = tree->left;
    while (cur != NULL && val != cur->data) {
        parent = cur;
        cur = (val < cur->data) ? cur->left : cur->right;
    }
    if (cur == NULL) {
        printf("\nThe value to be deleted is not present in the tree");
        return (tree);
    }
    if (cur->left == NULL)
        ptr = cur->right;
    else if (cur->right == NULL)
        ptr = cur->left;
    else {
        // Find the in–order successor and its parent
        psuc = cur;
        cur = cur->left;
        while (suc->left != NULL) {
            psuc = suc;
            suc = suc->left;
        }
        if (cur == psuc) {
            // Situation 1
            suc->left = cur->right;
        }
        else {
            // Situation 2
            suc->left = cur->left;
            psuc->left = suc->right;
```

ANVITA KUMAR
C-22
Roll No.: 2104097

```
            suc->right = cur->right;
        }
        ptr = suc;
    }
    // Attach ptr to the parent node
    if (parent->left == cur)
        parent->left = ptr;
    else
        parent->right = ptr;
    free(cur);
    return tree;
}
struct node *search(struct node *tree, int  val)
{
    if(tree==NULL) {
        printf("\nThe tree is empty");
    }
    else if(val > tree->data)
        tree=tree->right;
    else if(val < tree->data)
        tree=tree->left;
    else
        return tree;
}
void preorderTraversal(struct node *tree)
{
    if (tree != NULL) {
        printf("%d\t", tree->data);
        preorderTraversal(tree->left);
        preorderTraversal(tree->right);
    }
```

ANVITA KUMAR
C-22
Roll No.: 2104097

```
}
void inorderTraversal(struct node *tree)
{
   if (tree != NULL) {
      inorderTraversal(tree->left);
      printf("%d\t", tree->data);
      inorderTraversal(tree->right);
   }
}
void postorderTraversal(struct node *tree)
{
   if (tree != NULL) {
      postorderTraversal(tree->left);
      postorderTraversal(tree->right);
      printf("%d\t", tree->data);
   }
}
int totalNodes(struct node *tree)
{
   if (tree == NULL)
      return 0;
   else
      return (totalNodes(tree->left) + totalNodes(tree->right) + 1);
}
int totalLeafNodes(struct node *tree)
{
   if (tree == NULL)
      return 0;
   else if ((tree->left == NULL) && (tree->right == NULL))
      return 1;
   else
```

ANVITA KUMAR
C-22
Roll No.: 2104097

```
    return (totalLeafNodes(tree->left) + totalLeafNodes(tree->right));
}
int totalInternalNodes(struct node *tree)
{
  if ((tree == NULL) || ((tree->left == NULL) && (tree->right == NULL)))
    return 0;
  else
    return (totalInternalNodes(tree->left) + totalInternalNodes(tree->right) + 1);
}
int Height(struct node *tree)
{
  int leftheight, rightheight;
  if (tree == NULL) return 0;
  else
  {
    leftheight = Height(tree->left);
    rightheight = Height(tree->right);
    if (leftheight > rightheight)
      return (leftheight + 1);
    else
      return (rightheight + 1);
  }
}
```

ANVITA KUMAR
C-22
Roll No.: 2104097

```
PS C:\Users\Saniha Kumar\Desktop\Anvita\C program\exp2\.vscode> cd "c:\Users\Saniha Kumar\Desktop\Anvita\C program\exp2\.vscode\" ;
if ($?) { gcc exp8.c -o exp8 } ; if ($?) { .\exp8 }

***List Of Operations***
1. Insertion
2. Deletion
3. Searching
4. Pre-order Traversal
5. In-order Traversal
6. Postorder Traversal
7. Total number of nodes
8. Total number of leaf nodes
9. Total number of internal nodes
10. Find height of the tree
11. Exit
Enter your option : 1
Enter the value to be inserted: 11

***List Of Operations***
1. Insertion
2. Deletion
3. Searching
```

```
Enter your option : 1
Enter the value to be inserted: 9

***List Of Operations***
1. Insertion
2. Deletion
3. Searching
4. Pre-order Traversal
5. In-order Traversal
6. Postorder Traversal
7. Total number of nodes
8. Total number of leaf nodes
9. Total number of internal nodes
10. Find height of the tree
11. Exit
Enter your option : 1
Enter the value to be inserted: 8

***List Of Operations***
1. Insertion
2. Deletion
3. Searching
```

```
Enter your option : 1
Enter the value to be inserted: 13

***List Of Operations***
1. Insertion
2. Deletion
3. Searching
4. Pre-order Traversal
5. In-order Traversal
6. Postorder Traversal
7. Total number of nodes
8. Total number of leaf nodes
9. Total number of internal nodes
10. Find height of the tree
11. Exit
Enter your option : 1
Enter the value to be inserted: 12

***List Of Operations***
1. Insertion
2. Deletion
3. Searching
```

ANVITA KUMAR
C-22
Roll No.: 2104097

```
Enter your option : 1
Enter the value to be inserted: 15

***List Of Operations***
1. Insertion
2. Deletion
3. Searching
4. Pre-order Traversal
5. In-order Traversal
6. Postorder Traversal
7. Total number of nodes
8. Total number of leaf nodes
9. Total number of internal nodes
10. Find height of the tree
11. Exit
Enter your option : 1
Enter the value to be inserted: 3

***List Of Operations***
1. Insertion
2. Deletion
3. Searching
```

```
Enter your option : 2
Enter the element to be deleted: 3

***List Of Operations***
1. Insertion
2. Deletion
3. Searching
4. Pre-order Traversal
5. In-order Traversal
6. Postorder Traversal
7. Total number of nodes
8. Total number of leaf nodes
9. Total number of internal nodes
10. Find height of the tree
11. Exit
Enter your option : 3
Enter the element to be searched: 11
The value 11 is found in the tree
***List Of Operations***
1. Insertion
2. Deletion
3. Searching
```

```
Enter your option : 4
The elements of the tree are :
11      9       8       13      12      15
***List Of Operations***
1. Insertion
2. Deletion
3. Searching
4. Pre-order Traversal
5. In-order Traversal
6. Postorder Traversal
7. Total number of nodes
8. Total number of leaf nodes
9. Total number of internal nodes
10. Find height of the tree
11. Exit
Enter your option : 5
The elements of the tree are :
8       9       11      12      13      15
***List Of Operations***
1. Insertion
2. Deletion
3. Searching
```

ANVITA KUMAR
C-22
Roll No.: 2104097

```
Enter your option : 6
The elements of the tree are :
8       9       12      15      13      11
***List Of Operations***
1. Insertion
2. Deletion
3. Searching
4. Pre-order Traversal
5. In-order Traversal
6. Postorder Traversal
7. Total number of nodes
8. Total number of leaf nodes
9. Total number of internal nodes
10. Find height of the tree
11. Exit
Enter your option : 7
Total no. of nodes = 6
***List Of Operations***
1. Insertion
2. Deletion
3. Searching
4. Pre-order Traversal
```

```
Enter your option : 8
Total no. of leaf nodes = 3
***List Of Operations***
1. Insertion
2. Deletion
3. Searching
4. Pre-order Traversal
5. In-order Traversal
6. Postorder Traversal
7. Total number of nodes
8. Total number of leaf nodes
9. Total number of internal nodes
10. Find height of the tree
11. Exit
Enter your option : 9
Total no. of internal nodes = 3
***List Of Operations***
1. Insertion
2. Deletion
3. Searching
4. Pre-order Traversal
5. In-order Traversal
```

```
8. Total number of leaf nodes
9. Total number of internal nodes
10. Find height of the tree
11. Exit
Enter your option : 10
The height of the tree = 3
***List Of Operations***
1. Insertion
2. Deletion
3. Searching
4. Pre-order Traversal
5. In-order Traversal
6. Postorder Traversal
7. Total number of nodes
8. Total number of leaf nodes
9. Total number of internal nodes
10. Find height of the tree
11. Exit
Enter your option : 11

        EXIT POINT!
PS C:\Users\Saniha Kumar\Desktop\Anvita\C program\exp2\.vscode>
```