Program :

#include <stdio.h>

#include <stdlib.h>

struct node {

    int data;

    struct node \*left;

    struct node \*right;

};

struct node\* insert(struct node\* node, int data)

{

    if (node == NULL)

    {

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

        node->data = data;

        node->left = NULL;

        node->right = NULL;

    }

    else if (data < node->data)

    {

        node->left = insert(node->left, data);

    }

    else if (data > node->data)

    {

        node->right = insert(node->right, data);

    }

    return node;

}

struct node \*delete(struct node \*root, int data)

{

    if (root == NULL)

    {

        return root;

    }

    else if (data < root->data)

    {

        root->left = delete(root->left, data);

    }

    else if (data > root->data)

    {

        root->right = delete(root->right, data);

    }

    else

    {

        if (root->left == NULL)

        {

            struct node \*temp = root->right;

            free(root);

            return temp;

        }

        else if (root->right == NULL)

        {

            struct node \*temp = root->left;

            free(root);

            return temp;

        }

        struct node \*temp = root->right;

        while (temp->left != NULL)

        {

            temp = temp->left;

        }

        root->data = temp->data;

        root->right = delete(root->right, temp->data);

    }

    return root;

}

struct node \*search(struct node \*root, int data)

{

    if (root == NULL)

    {

        return NULL;

    }

    else if (data == root->data)

    {

        return root;

    }

    else if (data < root->data)

    {

        return search(root->left, data);

    }

    else

    {

          return search(root->right, data);

    }

}

void inorder(struct node\* node)

{

    if(node==NULL)

     return;

    inorder(node->left);

    printf("%d ",node->data);

    inorder(node->right);

}

int main()

{

    struct node \*root = NULL;

    int choice, data,value;

        printf("\nList of operations in Binary Search Tree\n");

        printf("1. Insert a node");

        printf(" 2. Delete a node");

        printf(" 3. Search a node");

        printf(" 4. Inorder Traversal (ascending node)");

        printf(" 5. Exit\n");

    while (1)

    {

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice)

        {

            case 1:

                do

                   {

                     printf("Enter data to be inserted : ");

                     scanf("%d", &value);

                     root = insert(root, value);

                     printf("Enter 0 to exit and 1 to insert another node :  ");

                     scanf(" %d", &choice);

                   } while (choice == 1 || choice == 1);

                break;

            case 2:

                printf("Enter the element to be deleted: ");

                scanf("%d", &data);

                root = delete(root, data);

                printf("Element %d has been deleted successfully!\n", data);

                break;

            case 3:

                printf("Enter the element to be searched: ");

                scanf("%d", &data);

                struct node \*searchResult = search(root, data);

                if (searchResult != NULL)

                {

                    printf("Element %d is present in the Binary Search Tree\n", data);

                }

                else

                {

                    printf("Element %d is not present in the Binary Search Tree\n", data);

                }

                break;

            case 4:

                inorder(root);

                printf("\n");

                break;

            case 5:

                printf("EXIT");

                break;

        }

    }

    return 0;

}

Output :

PS C:\Users\HP\Desktop\coding> cd "c:\Users\HP\Desktop\coding\C\" ; if ($?) { gcc bst.c -o bst } ; if ($?) { .\bst }

List of operations in Binary Search Tree

1. Insert a node 2. Delete a node 3. Search a node 4. Inorder Traversal (ascending node) 5. Exit

Enter your choice: 1

Enter data to be inserted : 50

Enter 0 to exit and 1 to insert another node : 1

Enter data to be inserted : 40

Enter 0 to exit and 1 to insert another node : 1

Enter data to be inserted : 60

Enter 0 to exit and 1 to insert another node : 1

Enter data to be inserted : 10

Enter 0 to exit and 1 to insert another node : 1

Enter data to be inserted : 45

Enter 0 to exit and 1 to insert another node : 1

Enter data to be inserted : 75

Enter 0 to exit and 1 to insert another node : 0

Enter your choice: 4

10 40 45 50 60 75

Enter your choice: 3

Enter the element to be searched: 75

Element 75 is present in the Binary Search Tree

Enter your choice: 2

Enter the element to be deleted: 40

Element 40 has been deleted successfully!

Enter your choice: 4

10 45 50 60 75

Enter your choice: 5

EXIT