EXPERIMENT-9

Program :

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

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

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

struct Node* insert(struct Node* root, int value) {
    if (root == NULL)
        return createNode(value);

    if (value < root->data)
        root->left = insert(root->left, value);
    else if (value > root->data)
        root->right = insert(root->right, value);
    return root;
}

struct Node* search(struct Node* root, int key) {
    if (root == NULL || root->data == key)
        return root;

    if (key < root->data)
```

```
        return search(root→left, key);
    else
        return search(root→right, key);
}

struct Node* findMin(struct Node* node) {
    while (node && node→left ≠ NULL)
        node = node→left;
    return node;
}

struct Node* deleteNode(struct Node* root, int key) {
    if (root == NULL)
        return root;

    if (key < root→data) {
        root→left = deleteNode(root→left, key);
    } else if (key > root→data) {
        root→right = deleteNode(root→right, key);
    } 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 = findMin(root→right);
        root→data = temp→data;
        root→right = deleteNode(root→right, temp→data);
    }

    return root;
}
```

```
void inorder(struct Node* root) {
    if (root != NULL) {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}

void freeTree(struct Node* root) {
    if (root != NULL) {
        freeTree(root->left);
        freeTree(root->right);
        free(root);
    }
}

int main() {
    struct Node* root = NULL;
    int choice, value;

    while (1) {
        printf("\n--- Binary Search Tree Menu ---\n");
        printf("1. Insert\n");
        printf("2. Search\n");
        printf("3. Delete\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
        case 1:
            printf("Enter value to insert: ");
            scanf("%d", &value);
            root = insert(root, value);
            printf("Inserted.\n");
            printf("Current tree (Inorder): ");
```

```
            inorder(root);
            printf("\n");
            break;
        case 2:
            printf("Enter value to search: ");
            scanf("%d", &value);
            if (search(root, value))
                printf("%d found in the tree.\n", value);
            else
                printf("%d not found.\n", value);
            printf("Current tree (Inorder): ");
            inorder(root);
            printf("\n");
            break;
        case 3:
            printf("Enter value to delete: ");
            scanf("%d", &value);
            root = deleteNode(root, value);
            printf("Deleted if existed.\n");
            printf("Current tree (Inorder): ");
            inorder(root);
            printf("\n");
            break;
        case 4:
            freeTree(root);
            printf("Exiting.\n");
            exit(0);
        default:
            printf("Invalid choice. Try again.\n");
        }
    }

    return 0;
}
```

Output :

```
cseb2@sjcet-OptiPlex-SFF-7020:~$ cd Alwin
cseb2@sjcet-OptiPlex-SFF-7020:~/Alwin$ gcc binary.c
cseb2@sjcet-OptiPlex-SFF-7020:~/Alwin$ ./a.out

--- Binary Search Tree Menu ---
1. Insert
2. Search
3. Delete
4. Exit
Enter your choice: 1
Enter value to insert: 50
Inserted.
Current tree (Inorder): 50

--- Binary Search Tree Menu ---
1. Insert
2. Search
3. Delete
4. Exit
Enter your choice: 1
Enter value to insert: 30
Inserted.
Current tree (Inorder): 30 50

--- Binary Search Tree Menu ---
1. Insert
2. Search
3. Delete
4. Exit
Enter your choice: 1
Enter value to insert: 70
Inserted.
Current tree (Inorder): 30 50 70

--- Binary Search Tree Menu ---
1. Insert
2. Search
3. Delete
4. Exit
Enter your choice: 2
Enter value to search: 30
30 found in the tree.
Current tree (Inorder): 30 50 70

--- Binary Search Tree Menu ---
1. Insert
2. Search
3. Delete
4. Exit
Enter your choice: 3
Enter value to delete: 50
Deleted if existed.
Current tree (Inorder): 30 70

--- Binary Search Tree Menu ---
1. Insert
2. Search
3. Delete
4. Exit
Enter your choice: 4
Exiting.
cseb2@sjcet-OptiPlex-SFF-7020:~/Alwin$ ^[[2~
```