//Implemention of various operation using binary search Tree

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

```
}
void preorder(struct Node* root)
  if (root != NULL)
     printf("%d ", root->data);
     preorder(root->left);
     preorder(root->right);
  }
}
void postorder(struct Node* root)
  if (root != NULL)
     postorder(root->left);
     postorder(root->right);
     printf("%d ", root->data);
  }
struct Node* findMin(struct Node* root)
  while (root && root->left != NULL)
     root = root->left;
  return root;
}
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 display(struct Node* root, int space)
  int i;
  if (root == NULL)
     return;
  space += 10;
  display(root->right, space);
  printf("\n");
  for (i = 10; i < \text{space}; i++)
     printf(" ");
  printf("%d\n", root->data);
  display(root->left, space);
}
int main()
  struct Node* root = NULL;
  int choice, value, key;
  while (1)
  {
     printf("\nBinary Search Tree Operations Menu:");
     printf("\n1. Insert a Node");
     printf("\n2. Delete a Node");
     printf("\n3. Search for a Node");
     printf("\n4. Inorder Traversal");
     printf("\n5. Preorder Traversal");
     printf("\n6. Postorder Traversal");
     printf("\n7. Display Tree");
     printf("\n8. Exit");
     printf("\nEnter your choice: ");
     scanf("%d", &choice);
     switch (choice)
        case 1:
          printf("Enter value to insert: ");
          scanf("%d", &value);
          root = insert(root, value);
          break;
        case 2:
          printf("Enter value to delete: ");
          scanf("%d", &value);
          root = deleteNode(root, value);
```

```
break;
     case 3:
        printf("Enter value to search: ");
        scanf("%d", &key);
        if (search(root, key) != NULL)
          printf("Node found!\n");
        else
          printf("Node not found!\n");
        break;
     case 4:
        printf("Inorder Traversal: ");
        inorder(root);
        printf("\n");
        break;
     case 5:
        printf("Preorder Traversal: ");
        preorder(root);
        printf("\n");
        break;
     case 6:
        printf("Postorder Traversal: ");
        postorder(root);
        printf("\n");
        break;
     case 7:
        printf("Displaying Tree Structure:\n");
        display(root, 0);
        break;
     case 8:
        exit(0);
     default:
        printf("Invalid choice! Please enter a valid option.\n");
  }
return 0;
```

}

OUTPUT:

```
Output
                                                                                                                                    Clear
         1 #include<stdio.h>
                                                                         Binary Search Tree Operations Menu:
P
         2 #include<stdlib.h>
                                                                         1. Insert a Node
         3 struct Node
                                                                         2. Delete a Node
4 - {
                                                                         3. Search for a Node
         5
               int data:
                                                                         4. Inorder Traversal
               struct Node* left;
         6
ఠ
                                                                         5. Preorder Traversal
               struct Node* right;
                                                                         6. Postorder Traversal
         8 };
 $
                                                                         7. Display Tree
        9 struct Node* createNode(int value)
        10 - {
0
                                                                         Enter your choice: 1
               struct Node* newNode = (struct Node*)malloc(sizeof(struct
                                                                         Enter value to insert: 7
                  Node));
(
        12
               newNode->data = value:
                                                                         Binary Search Tree Operations Menu:
               newNode->left = newNode->right = NULL;
        13
                                                                         1. Insert a Node
               return newNode;
(
                                                                         2. Delete a Node
                                                                         3. Search for a Node
        16 struct Node* insert(struct Node* root, int value)
                                                                         4. Inorder Traversal
 JS
        17 - {
                                                                         5. Preorder Traversal
                                         C & Share Run
       main.c
                                                                          Output
4
                                                                         Enter value to insert: 5
        1 #include<stdio.h>
R
        2 #include<stdlib.h>
                                                                         Binary Search Tree Operations Menu:
         3 struct Node
                                                                         1. Insert a Node
4 - {
                                                                         2. Delete a Node
                                                                         3. Search for a Node
               struct Node* left;
9
                                                                         4. Inorder Traversal
               struct Node* right;
                                                                         5. Preorder Traversal
         8 };
 É
        9 struct Node* createNode(int value)
                                                                         6. Postorder Traversal
                                                                         7. Display Tree
        10 - {
0
               struct Node* newNode = (struct Node*)malloc(sizeof(struct
        11
                                                                         Enter your choice: 7
                  Node));
(3)
        12
               newNode->data = value;
                                                                         Displaying Tree Structure:
        13
               newNode->left = newNode->right = NULL;
(
        14
               return newNode;
        15 }
        16 struct Node* insert(struct Node* root, int value)
 JS
                                         Output
ð
       main.c
                                                                         U. FUSCULUEL ITAVELSAL
        1 #include<stdio.h>
                                                                         7. Display Tree
R
         2 #include<stdlib.h>
                                                                         8. Exit
         3 struct Node
                                                                         Enter your choice: 3
Enter value to search: 6
               int data;
                                                                         Node not found!
               struct Node* left:
ఠ
         6
               struct Node* right;
                                                                         Binary Search Tree Operations Menu:
                                                                         1. Insert a Node
 É
         9 struct Node* createNode(int value)
                                                                         2. Delete a Node
        10 - {
                                                                         3. Search for a Node
0
               struct Node* newNode = (struct Node*)malloc(sizeof(struct
        11
                                                                         4. Inorder Traversal
                  Node));
                                                                         5. Preorder Traversal
(
        12
               newNode->data = value;
                                                                         6. Postorder Traversal
        13
               newNode->left = newNode->right = NULL;
                                                                         7. Display Tree
        14
               return newNode:
(
                                                                         8. Exit
        15 }
                                                                         Enter your choice: 4
        16 struct Node* insert(struct Node* root, int value)
                                                                         Inorder Traversal: 5 7
       17 - {
                                         Output
4
                                                                         O. EXIL
        1 #include<stdio.h>
                                                                         Enter your choice: 5
R
         2 #include<stdlib.h>
                                                                         Preorder Traversal: 7 5
         3 struct Node
         4 - {
Binary Search Tree Operations Menu:
        5
               int data:
                                                                         1. Insert a Node
               struct Node* left;
         6
ਰ
                                                                            Delete a Node
               struct Node* right;
                                                                         3. Search for a Node
 鱼
                                                                         4. Inorder Traversal
        9 struct Node* createNode(int value)
                                                                         5. Preorder Traversal
        10 - {
                                                                         6. Postorder Traversal
0
              struct Node* newNode = (struct Node*)malloc(sizeof(struct
        11
                                                                         7. Display Tree
                  Node));
                                                                         8. Exit
(
        12
               newNode->data = value;
                                                                         Enter your choice: 1
        13
               newNode->left = newNode->right = NULL;
                                                                         Enter value to insert: 8
        14
               return newNode;
(
        15 }
                                                                         Binary Search Tree Operations Menu:
        16 struct Node* insert(struct Node* root, int value)
                                                                         1. Insert a Node
                                                                         Delete a Node
```

