```c
//Binary Search Tree
//Om Dattatray Gavande
//Class:-SY CSE A
//Roll No:-CS2145
#include <stdio.h>
#include <stdlib.h>
struct node {
  int data;
  struct node *left, *right;
};
struct node* newNode(int val) {
  struct node* n = (struct node*)malloc(sizeof(struct node));
  if (!n) {
     printf("Memory allocation failed\n");
     return NULL;
  }
  n->data = val;
  n->left = n->right = NULL;
  return n;
}
struct node* insertBST(struct node* root, int val) {
  if (root == NULL) return newNode(val);
  if (val < root->data)
     root->left = insertBST(root->left, val);
  else if (val > root->data)
     root->right = insertBST(root->right, val);
  return root;
}
struct node* createTree(struct node* root) {
  char c;
  do {
     int x;
     printf("Enter data for the node: ");
     scanf("%d", &x);
     root = insertBST(root, x);
     printf("Do you want to add more nodes? (y/n): ");
     scanf(" %c", &c);
  } while (c=='y' || c=='Y');
  return root;
}
void displayInorder(struct node* root) {
  if (root == NULL) {
     printf("Tree is empty\n");
     return;
  }
  if (root->left) displayInorder(root->left);
  printf("%d ", root->data);
  if (root->right) displayInorder(root->right);
}
struct node* find_minimum(struct node* root) {
  if (root == NULL) return NULL;
  while (root->left != NULL) root = root->left;
  return root;
```

```
}
struct node* find_maximum(struct node* root) {
  if (root == NULL) return NULL;
  while (root->right != NULL) root = root->right;
  return root;
}
struct node* searchBST(struct node* root, int key) {
  while (root != NULL) {
     if (key == root->data) return root;
     else if (key < root->data) root = root->left;
     else root = root->right;
  }
  return NULL;
}
int main() {
  struct node* root = NULL;
  int choice, x;
  char cont;
  do {
     printf("\nBinary Search Tree\n");
     printf("1. Create Tree\n");
     printf("2. Display Tree (In-order)\n");
     printf("3. Find Maximum Node\n");
     printf("4. Find Minimum Node\n");
     printf("5. Search Node\n");
     printf("6. Exit\n");
     printf("Enter your choice: ");
     scanf("%d", &choice);
     switch (choice) {
        case 1:
           root = createTree(root);
           break;
        case 2:
           displayInorder(root);
           printf("\n");
           break;
        case 3: {
           struct node* mx = find_maximum(root);
           if (mx) printf("Maximum value in the tree: %d\n", mx->data);
           else printf("Tree is empty\n");
           break;
        }
        case 4: {
           struct node* mn = find_minimum(root);
           if (mn) printf("Minimum value in the tree: %d\n", mn->data);
           else printf("Tree is empty\n");
           break;
        }
        case 5:
           printf("Enter the value to search for: ");
           scanf("%d", &x);
           if (searchBST(root, x))
              printf("Node found: %d\n", x);
           else
```

```
            printf("Node not found\n");
        break;
      case 6:
        printf("Exiting...\n");
        return 0;
      default:
        printf("Invalid choice! Please try again.\n");
    }
    printf("Do you want to continue? (y/n): ");
    scanf(" %c", &cont);
  } while (cont=='y' || cont=='Y');
  return 0;
}
```