// program to implement BST

#include <stdio.h>

#include <stdlib.h>

struct TreeNode {

int data;

struct TreeNode\* left;

struct TreeNode\* right;

};

struct TreeNode\* createNode(int data) {

struct TreeNode\* newNode = (struct TreeNode\*)malloc(sizeof(struct TreeNode));

newNode->data = data;

newNode->left = newNode->right = NULL;

return newNode;

}

struct TreeNode\* insert(struct TreeNode\* root, int data) {

if (root == NULL) {

return createNode(data);

}

if (data < root->data) {

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

} else if (data > root->data) {

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

}

return root;

}

int findMinimum(struct TreeNode\* root) {

if (root == NULL) {

printf("The tree is empty.\n");

return -1;

}

while (root->left != NULL) {

root = root->left;

}

return root->data;

}

int findMaximum(struct TreeNode\* root) {

if (root == NULL) {

printf("The tree is empty.\n");

return -1;

}

while (root->right != NULL) {

root = root->right;

}

return root->data;

}

int search(struct TreeNode\* root, int key) {

if (root == NULL) {

return 0;

}

if (root->data == key) {

return 1;

}

if (key < root->data) {

return search(root->left, key);

} else {

return search(root->right, key);

}

}

int main() {

struct TreeNode\* root = NULL;

int choice, data;

while (1) {

printf("\nBinary Search Tree Menu\n");

printf("1. Insert\n");

printf("2. Find Maximum\n");

printf("3. Find Minimum\n");

printf("4. Search\n");

printf("5. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

printf("Enter value to insert: ");

scanf("%d", &data);

root = insert(root, data);

break;

case 2:

printf("Maximum value: %d\n", findMaximum(root));

break;

case 3:

printf("Minimum value: %d\n", findMinimum(root));

break;

case 4:

printf("Enter value to search: ");

scanf("%d", &data);

if (search(root, data)) {

printf("%d found in the tree.\n", data);

} else {

printf("%d not found in the tree.\n", data);

}

break;

case 5:

exit(0);

default:

printf("Invalid choice. Please try again.\n");

}

}

return 0;

}

/\* Output:

Binary Search Tree Menu

1. Insert

2. Find Maximum

3. Find Minimum

4. Search

5. Exit

Enter your choice: 1

Enter value to insert: 32

Binary Search Tree Menu

1. Insert

2. Find Maximum

3. Find Minimum

4. Search

5. Exit

Enter your choice: 1

Enter value to insert: 50

Binary Search Tree Menu

1. Insert

2. Find Maximum

3. Find Minimum

4. Search

5. Exit

Enter your choice: 1

Enter value to insert: 10

Binary Search Tree Menu

1. Insert

2. Find Maximum

3. Find Minimum

4. Search

5. Exit

Enter your choice: 2

Maximum value: 50

Binary Search Tree Menu

1. Insert

2. Find Maximum

3. Find Minimum

4. Search

5. Exit

Enter your choice: 3

Minimum value: 10

Binary Search Tree Menu

1. Insert

2. Find Maximum

3. Find Minimum

4. Search

5. Exit

Enter your choice: 4

Enter value to search: 10

10 found in the tree.

Binary Search Tree Menu

1. Insert

2. Find Maximum

3. Find Minimum

4. Search

5. Exit

Enter your choice: 5

Process returned 0 (0x0) execution time : 32.331 s

Press any key to continue. \*/