

Experiment 4: Write a C program that uses functions to perform the following: a) Create a binary search tree of characters.

```
experiment4a.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  typedef struct Node {
4      char data;
5      struct Node* left;
6      struct Node* right;
7  } Node;
8  Node* createNode(char data) {
9      Node* newNode = (Node*)malloc(sizeof(Node));
10     if (newNode == NULL) {
11         printf("Memory error\n");
12         return NULL;
13     }
14     newNode->data = data;
15     newNode->left = NULL;
16     newNode->right = NULL;
17     return newNode;
18 }
19 Node* insert(Node* root, char data) {
20     if (root == NULL) {
21         return createNode(data);
22     }
23     if (data < root->data) {
24         root->left = insert(root->left, data);
25     } else if (data > root->data) {
26         root->right = insert(root->right, data);
27     }
28     return root;
29 }
30 Node* search(Node* root, char data) {
31     if (root == NULL || root->data == data) {
32         return root;
33     }
34     if (data < root->data) {
35         return search(root->left, data);
36     } else {
37         return search(root->right, data);
38     }
39 }
40 void inorderTraversal(Node* root) {
41     if (root != NULL) {
42         inorderTraversal(root->left);
43         printf("%c ", root->data);
44         inorderTraversal(root->right);
45     }
46 }
47 }
```

```

48 int main() {
49     Node* root = NULL;
50     root = insert(root, 'F');
51     insert(root, 'D');
52     insert(root, 'J');
53     insert(root, 'B');
54     insert(root, 'E');
55     insert(root, 'G');
56     insert(root, 'K');
57     printf("Inorder traversal of the binary search tree: ");
58     inorderTraversal(root);
59     printf("\n");
60     char searchChar = 'E';
61     Node* result = search(root, searchChar);
62     if (result != NULL) {
63         printf("Character '%c' found in the tree.\n", searchChar);
64     } else {
65         printf("Character '%c' not found in the tree.\n", searchChar);
66     }
67     free(root);
68     return 0;
69 }

```

Output:

```

D:\SubhamDasDSUC\experim X + v
Inorder traversal of the binary search tree: B D E F G J K
Character 'E' found in the tree.

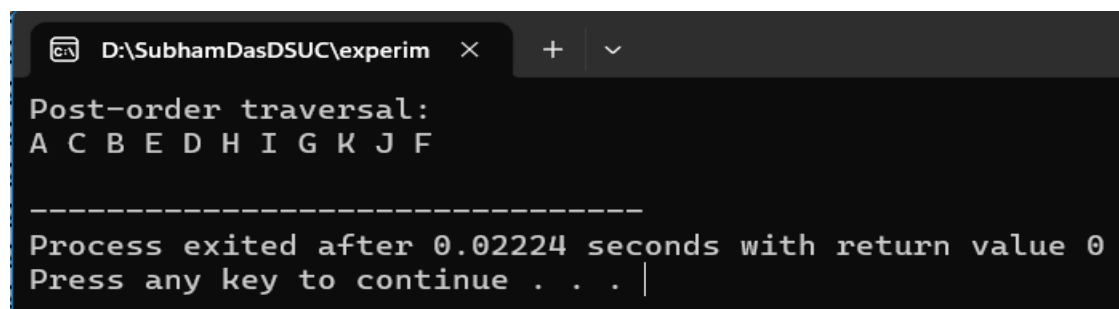
-----
Process exited after 0.02791 seconds with return value 0
Press any key to continue . . . |

```

b) Traverse the above Binary search tree recursively in Post order.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  typedef struct TreeNode {
4      char data;
5      struct TreeNode *left;
6      struct TreeNode *right;
7  } TreeNode;
8  TreeNode *createNode(char data) {
9      TreeNode *newNode = (TreeNode *)malloc(sizeof(TreeNode));
10     newNode->data = data;
11     newNode->left = newNode->right = NULL;
12     return newNode;
13 }
14 TreeNode *insert(TreeNode *root, char data) {
15     if (root == NULL) {
16         return createNode(data);
17     }
18     if (data < root->data) {
19         root->left = insert(root->left, data);
20     } else if (data > root->data) {
21         root->right = insert(root->right, data);
22     }
23     return root;
24 }
25 void postOrderTraversal(TreeNode *root) {
26     if (root == NULL)
27         return;
28     postOrderTraversal(root->left);
29     postOrderTraversal(root->right);
30     printf("%c ", root->data);
31 }
32 int main() {
33     int i;
34     TreeNode *root = NULL;
35     char charArray[] = {'F', 'D', 'J', 'B', 'E', 'G', 'K', 'A', 'C', 'I', 'H'};
36     int numChars = sizeof(charArray) / sizeof(charArray[0]);
37     for (i = 0; i < numChars; i++) {
38         root = insert(root, charArray[i]);
39     }
40     printf("Post-order traversal:\n");
41     postOrderTraversal(root);
42     printf("\n");
43     return 0;
44 }
```

Output:



```
D:\SubhamDasDSUC\experim  ×  +  ▾
Post-order traversal:
A C B E D H I G K J F
-----
Process exited after 0.02224 seconds with return value 0
Press any key to continue . . .
```