

Data Structures and Algorithms

Lab Report

Lab10



Group Members Name & Reg #:	<u>Muhammad Haris Irfan</u> (FA18-BCE-090)
Class	Data Structures and Algorithms CSC211 (BCE-3B)
Instructor's Name	Dilshad Sabir

In Lab Tasks

Task:1

Your Task is to complete the 'Node Deletion' Function.

Solution:

The code is shown below,

```
struct node* delete_node(struct node* root, int data)
{
    if (root == NULL){
        printf("\nNode not found!!");
        return root;
    }

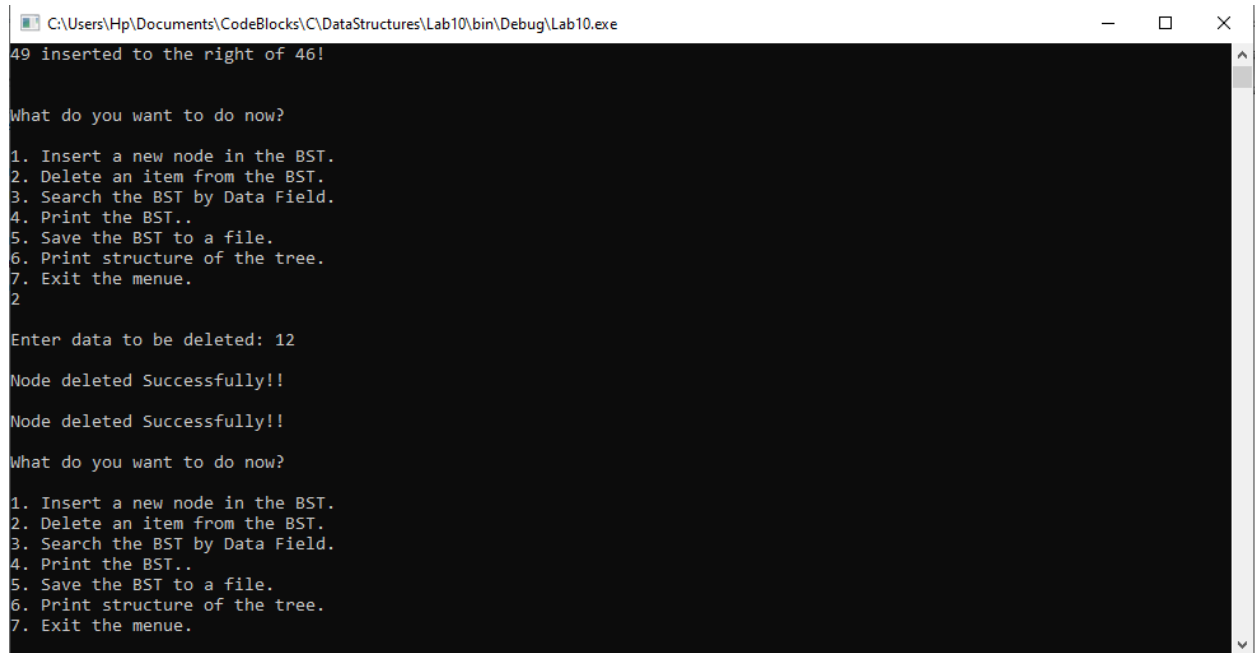
    if (data < root->data)
        root->left = delete_node(root->left, data);

    else if (data > root->data)
        root->right = delete_node(root->right, data);

    else{
        if (root->left == NULL){
            struct node *temp = root->right;
            free(root);
            printf("\nNode deleted Successfully!!\n");
            return temp;
        }
        else if (root->right == NULL){
            struct node *temp = root->left;
            free(root);
            printf("\nNode deleted Successfully!!\n");
            return temp;
        }

        struct node* temp = find_smallest_node(root->right);
        root->data = temp->data;
        root->right = delete_node(root->right, temp->data);
    }
    return root;
}
```

The Result of the following code is attached below:



```
C:\Users\Hp\Documents\CodeBlocks\C\DataStructures\Lab10\bin\Debug\Lab10.exe
49 inserted to the right of 46!

What do you want to do now?
1. Insert a new node in the BST.
2. Delete an item from the BST.
3. Search the BST by Data Field.
4. Print the BST..
5. Save the BST to a file.
6. Print structure of the tree.
7. Exit the menu.
2

Enter data to be deleted: 12

Node deleted Successfully!!

Node deleted Successfully!!

What do you want to do now?
1. Insert a new node in the BST.
2. Delete an item from the BST.
3. Search the BST by Data Field.
4. Print the BST..
5. Save the BST to a file.
6. Print structure of the tree.
7. Exit the menu.
```

=====

Task:2

Your Task is to complete the 'Node Search' Function.

Solution:

The code is shown below,

```

bool bst_search(struct node * root, int key)
{
    /** Complete this function */

    if(root->data==key)
        return true;
    if(root == NULL)
        return(false);
    while(1){
        if(key < root->data){
            root = root->left;
            if(root->data == key)
                return (true);
        }
        if(key > root->data){
            root = root->right;
            if(root->data == key)
                return (true);
        }
        if(root->left == NULL || root->right == NULL)
            return (false);
    }
}

```

The Result of the following code is attached below:

```

C:\Users\Hp\Documents\CodeBlocks\C\DataStructures\Lab10\bin\Debug\Lab10.exe
15 inserted to the left of 16!
10 inserted to the left of 12!
80 inserted to the right of 75!
49 inserted to the right of 46!

What do you want to do now?
1. Insert a new node in the BST.
2. Delete an item from the BST.
3. Search the BST by Data Field.
4. Print the BST..
5. Save the BST to a file.
6. Print structure of the tree.
7. Exit the menu.
3

Enter data to search: 12

Search key found in the tree!
What do you want to do now?
1. Insert a new node in the BST.
2. Delete an item from the BST.
3. Search the BST by Data Field.
4. Print the BST..
5. Save the BST to a file.
6. Print structure of the tree.
7. Exit the menu.

```

=====

Task:3

Your Task is to complete the 'Pre-Order' and 'Post-Order' Printing Functions.

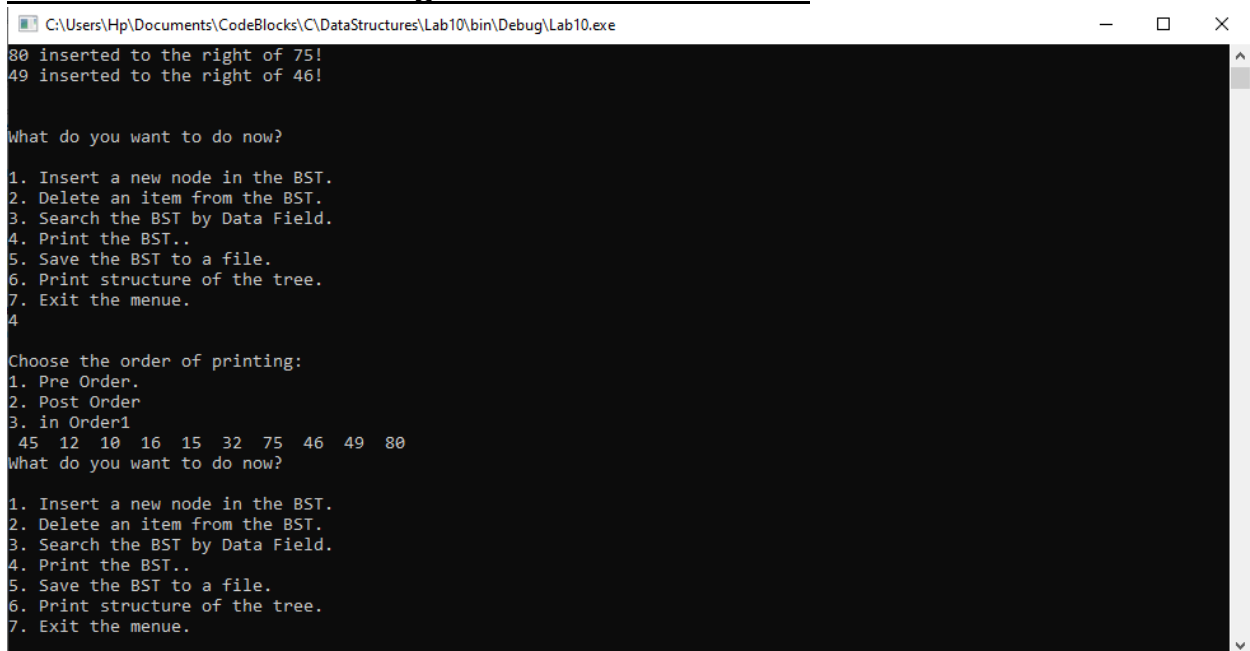
Solution:

The code is shown below,

Pre-Order

```
void print_pre_order(struct node * root)
{
    /** Complete this function */
    if(root == NULL)
        printf("The tree is empty!!");
    if(root !=NULL)
        printf(" %d ",root->data);
    if(root->left != NULL){
        print_pre_order(root->left);
    }
    if(root->right != NULL){
        print_pre_order(root->right);
    }
}
```

The Result of the following code is attached below:



```
C:\Users\Hp\Documents\CodeBlocks\C\DataStructures\Lab10\bin\Debug\Lab10.exe
80 inserted to the right of 75!
49 inserted to the right of 46!

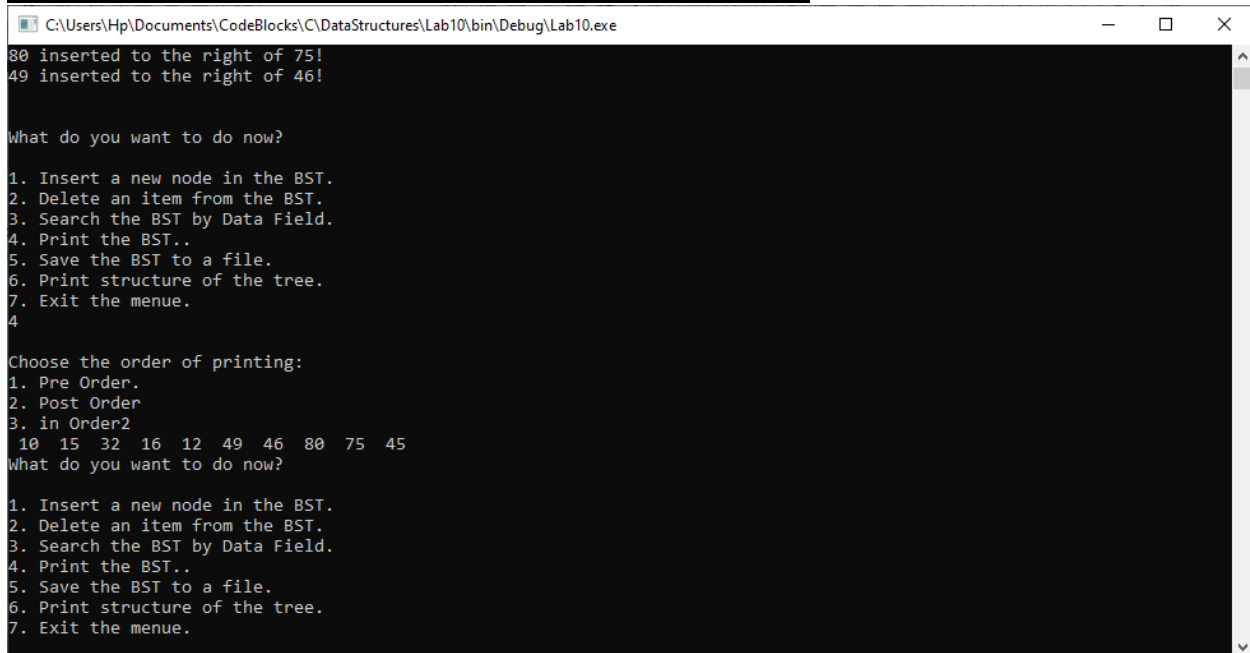
What do you want to do now?
1. Insert a new node in the BST.
2. Delete an item from the BST.
3. Search the BST by Data Field.
4. Print the BST..
5. Save the BST to a file.
6. Print structure of the tree.
7. Exit the menu.
4

Choose the order of printing:
1. Pre Order.
2. Post Order
3. in Order1
45 12 10 16 15 32 75 46 49 80
What do you want to do now?
1. Insert a new node in the BST.
2. Delete an item from the BST.
3. Search the BST by Data Field.
4. Print the BST..
5. Save the BST to a file.
6. Print structure of the tree.
7. Exit the menu.
4
```

Post-Order

```
void print_post_order(struct node * root)
{
    /** Complete this function */
    if(root == NULL)
        printf("The tree is empty!!");
    if(root->left != NULL){
        print_post_order(root->left);
    }
    if(root->right != NULL){
        print_post_order(root->right);
    }
    printf(" %d ", root->data);
}
```

The Result of the following code is attached below:



```
C:\Users\Hp\Documents\CodeBlocks\C\DataStructures\Lab10\bin\Debug\Lab10.exe
80 inserted to the right of 75!
49 inserted to the right of 46!

What do you want to do now?
1. Insert a new node in the BST.
2. Delete an item from the BST.
3. Search the BST by Data Field.
4. Print the BST..
5. Save the BST to a file.
6. Print structure of the tree.
7. Exit the menu.
4

Choose the order of printing:
1. Pre Order.
2. Post Order
3. in Order2
10 15 32 16 12 49 46 80 75 45
What do you want to do now?
1. Insert a new node in the BST.
2. Delete an item from the BST.
3. Search the BST by Data Field.
4. Print the BST..
5. Save the BST to a file.
6. Print structure of the tree.
7. Exit the menu.
4
```

=====

Post Lab Task.

Task 4:

Save the Tree data to a file (In-Order, Pre-Order and Post-Order)

Solution

The code is shown below,

In-Order

```
bool save_in_order(struct node * root, FILE * fptr)
{
    /** Complete this function */

    int data = root->data;
    if(root->left == NULL)    /// Now data of this node will be printed
        fprintf(fptr, " %d ", data);

    if(root->left != NULL)
    {
        save_in_order(root->left, fptr);
        fprintf(fptr, " %d ", data);
    }

    if(root->right != NULL)
    {
        save_in_order(root->right, fptr);
    }

    return;
}
```

Pre-Order

```
bool save_pre_order(struct node * root, FILE * fptr)
{
    int data = root->data;
    if(root != NULL)
        fprintf(fptr, " %d ", data);
    if(root->left != NULL){
        save_pre_order(root->left, fptr);
    }
    if(root->right != NULL){
        save_pre_order(root->right, fptr);
    }
    return;
}
```

Post-Order

```
bool save_post_order(struct node * root, FILE * fptr)
{
    int data = root->data;
    if(root->left != NULL){
        print_post_order(root->left);
    }
    if(root->right != NULL){
        print_post_order(root->right);
    }
    fprintf(fptr, " %d ", data);
    return;
}
```

=====

Task 5:

Load tree from a file containing numbers.

Solution

The code is shown below,

```
int load_tree(struct node * root, FILE * fptr)
{
    /** Complete this function */
    printf("\n\nPrinting tree loaded from a file.\n");
    int i;
    for(int j=0;j<10;j++){
        fscanf(fptr, "%d", &i);
        insert_node(&root, i);
    }
}
```

The Result of the following code is attached below:


```
C:\Users\Hp\Documents\CodeBlocks\C\DataStructures\Lab10\bin\Debug\Lab10.exe
Hello! This program lets you manage you an integer BST:
Do you want to load a tree from the file? (y/n): y

Printing tree loaded from a file.

1969897746 as Root node inserted!
1969897746 inserted to the right of 1969897746!
1969897746 inserted to the right of 1969897746!
1969897746 inserted to the right of 1969897746!
1969897746 inserted to the right of 1969897746!
1969897746 inserted to the right of 1969897746!
1969897746 inserted to the right of 1969897746!
1969897746 inserted to the right of 1969897746!
1969897746 inserted to the right of 1969897746!
45 as Root node inserted!
12 inserted to the left of 45!
16 inserted to the right of 12!
32 inserted to the right of 16!
75 inserted to the right of 45!
46 inserted to the left of 75!
15 inserted to the left of 16!
10 inserted to the left of 12!
80 inserted to the right of 75!
49 inserted to the right of 46!

What do you want to do now?
```

Conclusion:

In this lab, we implemented the functions to search, delete and print the Binary Search Tree, furthermore in Post-lab we implemented saving the Binary Search Tree as well as loading a tree from a file containing numbers.

THE END