Green University of Bangladesh

Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering

Semester: (Summer, Year:2022), B.Sc. in CSE (Day)

LAB REPORT NO # 06

Course Title: Data Structure Lab

Course Code: CSE 106 Section: PC-213DA

Student Details

Name	ID
Pankaj Mahanto	213902002

Lab Date	**: 17/08/2022**
Submission Date	**: 19/08/2022**
Course Teacher's Name	**: Farhana Akter Sunny**

Farhana Akter Sunny

Senior Professor

Green University of Bangladesh

[For Teachers use only: Don't Write anything inside the box]

Lab Report Status

Marks: . . . . **Signature: . . . .**

Comments: . . . . **Date: . . . .**

1. TITLE OF THE LAB EXPERIMENT [1]

- Implement a C program that is able to BST Preoder,Inoder,Postorder and Input taken from the user.

2. OBJECTION [1]

In this problem I will discuss Binary Search Tree and how it uses?

3. PROCEDURE /ANALYSIS/DESIGN/PSEUDOCODE [2]

Algorithm for Inserting a node in a tree:

```
1 Create a new node and assign values to it. insert(node, key)

2 If root == NULL, return the new node to the calling function.

3 If root=>data < key Call the insert function with root=>right and assign the return value
in  root=>right. root->right = insert(root=>right, key)

4 if root=>data > key

5 call the insert function with root->left and assign the return value in root=>left.
root=>left =insert(root=>left, key)

6 Finally, return the original root pointer to the calling function.
```

Algorithm for Pre-order Traversal of a tree:

```
1 Visit root node.

2 Recursively traverse left sub-tree.

3 Recursively traverse right sub-tree.
```

Algorithm for Post-order Traversal of a tree:

```
1 Recursively traverse left sub-tree.

2 Recursively traverse right sub-tree.

3 Visit root node.
```

Algorithm to In-order BST:

1 Recursively traverse left sub-tree.

2 visit root node.

3 Recursively traverse right sub-tree.

4. IMPLEMENTATION

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node *left, *right;
};
struct node *new_node(int n)
{
    struct node *temp;
    temp = (struct node *)malloc(sizeof(struct node));
    temp->data = n;
    temp->left = temp->right = NULL;
    return temp;
}
struct node *insert(struct node *node, int value)
{
    if (node == NULL)
    {
        return new_node(value);
    }
    else
    {
        if (value < node->data)
        {
            node->left = insert(node->left, value);
        }
        else if (value > node->data)
        {
            node->right = insert(node->right, value);
        }
        return node;
    }
}
// A utility function to do inorder traversal of BST
void inorder(struct node *root)
{
```

```c
    if (root != NULL)
    {
        inorder(root->left);
        printf("%4d", root->data);
        inorder(root->right);
    }
}
// preorder code
void preorder(struct node *root)
{

    if (root != NULL)
    {
        printf("%4d", root->data);
        preorder(root->left);
        preorder(root->right);
    }
}
// postorder code
void postorder(struct node *root)
{

    if (root != NULL)
    {
        postorder(root->left);
        postorder(root->right);
        printf("%4d", root->data);
    }
}

int main()
{
    struct node *root = NULL;
    int n, i;
    printf("\nenter the terms terms:\n");
    scanf("%d", &n);
    int a[n];
    for (i = 0; i < n; i++)
    {
        printf("element-%d:", i + 1);
        scanf("%d", &a[i]);
    }

    for (i = 0; i < n; i++)
    {
        root = insert(root, a[i]);
    }
    while (1)
    {
        int n;
        printf("\n\t\t\t\t--------------------------------------------------
\t\t\t\t");
        printf("\n\t\t\t\t|              1.Pre_order                   |\t\t");
        printf("\n\t\t\t\t|              2.In_order                    |\t\t");
        printf("\n\t\t\t\t|              3.Post_order                  |\t\t");
        printf("\n\t\t\t\t|              0.Exit                        |\t\t");
        printf("\n\t\t\t\t--------------------------------------------------
\t\t\t\t");
        printf("\nEnter Your Choice:\n");
        scanf("%d", &n);
        switch (n)
        {
        case 1:
            printf("\n\n");
            printf("\nPre_order traversal BST:\n");
            preorder(root);
            break;

        case 2:
            printf("\n\n");
            printf("\nIn-order traversal BST:\n");
            inorder(root);
            break;

        case 3:
            printf("\n\n");
            printf("\nPost-order traversal BST:\n");
            postorder(root);
            break;
        case 0:
            exit(0);
            break;

        default:
            printf("\nInvalid Choice\n");
            break;
        }
    }

    return 0;
}
```

5. TEST RESULT

Output:

```
enter the terms terms:
7
element-1:50
element-2:20
element-3:30
element-4:40
element-5:60
element-6:70
element-7:80

                       ----------------------------------------------
                       |          1.Pre_order                  |
                       |          2.In_order                   |
                       |          3.Post_order                 |
                       |          0.Exit                       |
                       ----------------------------------------------
Enter Your Choice:
1



Pre-order traversal BST:
  50  20  30  40  60  70  80
                       ----------------------------------------------
                       |          1.Pre_order                  |
                       |          2.In_order                   |
                       |          3.Post_order                 |
                       |          0.Exit                       |
                       ----------------------------------------------
Enter Your Choice:
2



In-order traversal BST:
  20  30  40  50  60  70  80
                       ----------------------------------------------
                       |          1.Pre_order                  |
                       |          2.In_order                  |
                       |          3.Post_order                 |
                       |          0.Exit                    |
                       ----------------------------------------------
Enter Your Choice:
3



Post-order traversal BST:
  40  30  20  80  70  60  50
                       ----------------------------------------------
                       |          1.Pre_order                  |
                       |          2.In_order                  |
                       |          3.Post_order                 |
                       |          0.Exit                    |
                                ---------------------------------------------------
Enter Your Choice:
```

6. ANALYSIS AND DISCUSSION

In this problem we get the proper use of Binary Search Tree. How did it used in doubly link list? It is very similarity in doubly link list. In this problem we actually three types of work found in preorder,in-order and post-order. Create Binary Search Tree this is the main concept in this problem and insert also.