

## PART A

(PART A : TO BE REFERRED BY STUDENTS)

### Experiment No.04

#### A.1 Aim:

Implementation of operations on Binary Search Tree.

#### A.2 Prerequisite:

C programming

#### A.3 Outcome:

After successful completion of this experiment students will be,

Apply concepts of Trees and Graphs to a given problem.

#### A.4 Theory:

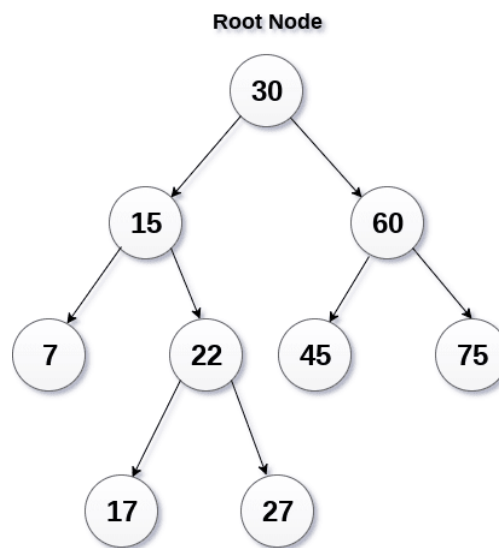
##### Binary Search Tree:

A binary search tree is an ordered binary tree. A Binary Search Tree (BST) is a tree in which all the nodes follow following properties –

All the nodes in the left sub-tree have a value less than that of the root (parent) node.

All the nodes in the right sub-tree have a value either equal to or greater than that of the root (parent) node.

These rules are applicable to every subtree in a tree.



**Binary Search Tree**

**Node Structure:**

```
struct BSTnode
{
    int data;
    struct BSTnode *left;
    struct BSTnode *right;
}
```

**Algorithm:****1. Create BST and Insert a node:**

Step 1: Call malloc to allocate memory to newnode

Step 2: Set newnode->data and newnode->left = newnode->right = NULL

Step 3: if root = NULL then

    root = temp = newnode  
    return root

    else

        repeat step 4 while temp != NULL

Step 4:           if newnode->data <= temp->data

                  if temp->left == NULL

                    temp->left = newnode

                    return root

                    temp = temp->left

                else

                  if temp->right == NULL

                    temp->right = newnode

                    return root

                    temp = temp->right

Step 5: Exit

**320517528. Display: (Inorder Traversal)**

Step 1: IF root != NULL then

Step 2: Inorder(root->left)

Step 3: Print root->data

Step 4: Inorder(root->right)

    [End of Loop]

Step 5: Stop

## PART B

(PART B : TO BE COMPLETED BY STUDENTS)

*(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Black board access available)*

Roll. No.71	Name: Omkar Gujja
Class: 2	Batch: MTRX
Date of Experiment: 6/3/2021	Date of Submission: 13/3/2021
Grade:	

### B.1 Software Code written by student:

```
#include<stdio.h>
#include<stdlib.h>

struct BSTnode {
    int data;
    struct BSTnode *left;
    struct BSTnode *right;
};

struct BSTnode* newNode(int item)
{
    struct BSTnode* temp
        = (struct BSTnode*)malloc(sizeof(struct BSTnode));
    temp->data = item;
    temp->left = temp->right = NULL;
    return temp;
}

struct BSTnode* insert(struct BSTnode* node, int data)
{
    if (node == NULL)
        return newNode(data);

    if (data < node->data)
```

```
    node->left = insert(node->left, data);
else if (data > node->data)
    node->right = insert(node->right, data);

return node;
}

void inorder(struct BSTnode* root)
{
    if (root != NULL) {
        inorder(root->left);
        printf("%d \n", root->data);
        inorder(root->right);
    }
}

int main()
{
    struct BSTnode* root = NULL;
    root = insert(root, 50);
    insert(root, 30);
    insert(root, 20);
    insert(root, 40);
    insert(root, 70);
    insert(root, 60);
    insert(root, 80);

    printf("Inorder traversal of BST : \n");
    inorder(root);

    return 0;
}
```

**B.2 Input and Output:**

```
Inorder traversal of BST :
20
30
40
50
60
70
80

...Program finished with exit code 0
Press ENTER to exit console.□
```

**B.3 Observations and learning:**

We observe how operations are performed on Binary search tree

**B.4 Conclusion:**

We have understood the implementation of Binary Tree.

**B.5 Question of Curiosity**

Q.1 What is binary tree and Binary search tree?

Soln.

IN BINARY TREE there is no ordering in terms of how the nodes are arranged

IN BINARY SEARCH TREE the left subtree has elements less than the nodes element and the right subtree has elements greater than the nodes element

Q.2 Write an algorithm to search an element in a binary search tree

soln.

Step 1: low = 0

Step2: high = n-1

Step 3: while (low <= high) repeat

step 4 through step 6 Step 4: mid = (low + high) / 2

Step 5: Is ( ele == a[mid]) then Loc = mid goto step 7 Otherwise

Step 6: is ( ele < a[mid])? then high = mid – 1 otherwise low = mid + 1 [end while – step 3]

Step 7: Is ( Loc >= 0 ) ? then Print “search element found at location “, loc

Otherwise Print “search element not found”.

Q.3 Construct a Binary search tree by inserting 43, 10, 79, 90, 12, 54, 11, 9, 50

```
Inorder traversal of BST :  
9  
10  
11  
12  
43  
50  
54  
79  
90  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```