	PUNE INSTITUTE OF COMPUTER TECHNOLOGY PUNE - 411043	
	Department of Electronics & Telecommunication	
	ASSESMENT YEAR: 2021-2022	CLASS: SE-5
	SUBJECT: DATA STRUCTURES	
EXPT No: 5	LAB Ref: SE/2021-22/	Starting date: 22/11/2021
	Roll No: 22108	Submission date: 29/11/2021
Title:	**Binary Search Tree Operations**	
Problem statement	Implement Binary search tree with operations Create, search, and recursive traversal.	
Prerequisites:	Basics of C programming	
	Decision making and loop controls	
	Choice based program	
Objectives:	Learn to create a Binary Search Tree (BST)	
	Implement various operation on BST to understand its effect on data.	
Theory:		

DS_Writeups_ETC

1) **Functions –**

- It is a self-contained block of statements that perform task of some kind.
- C program may have one or more functions
- C program must have at least one function i.e. main()
- There is no limit on number of functions
- Each function is called in a sequence specified by the function calls in main()
- After each function has done its job, control returns to next location from where it has been called...

2) **Tree** –

- A tree is defined as finite set of one or more nodes such that: 1) There is a specially designated node called as root.
  2) The remaining nodes are called as subtrees of the root.
- Leaf nodes – these are the terminal nodes of subtrees having pointer stored as 'NULL'.
- Node in this data structure will contain memory for two pointers for linking and rest for data. Pointers – one will be pointing to left subtree and other will be pointing to right subtree.
- Initializing a tree ❼ struct Node

```
{
    int data;
```

DS_Writeups_ETC

DS_Writeups_ETC

```
        struct node *left;
        struct node *right;
    }
```

C) Binary Search T

ERROR and REMEDY	-

DS_Writeups_ETC

Code

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

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

void end()
{
   printf("\n\tEnd of Execution!");
return;
}

struct node *create_node(int data)
{
   struct node *n;
   n = (struct node *)malloc(sizeof(struct
node));     n->data = data;     n->left = NULL;
   n->right = NULL;
return n;
}

void insert(struct node *root, int key)
{
   struct node *prev = NULL;
   int dupli = 0;
   while (root != NULL)
   {
      prev = root;
      if (key == root->data)
      {
         printf("Duplicate value not allowed, already present in the
tree!!");            dupli = 1;            end();
      }
      else if (key < root->data)
      {
         root = root->left;
      }
else
{
         root = root->right;
```

DS_Writeups_ETC

```
        }
    }
    struct node *new = create_node(key);
if (key < prev->data)
    {
        prev->left = new;
    }
else
    {
        prev->right = new;
    }
}

struct node *search(struct node *root, int key)
{     if (root ==
NULL)
    {
return 0;
    }
    if (root->data == key)
    {         return
root;
    }
    else if (root->data>key)
    {
        search(root->left, key);
    }
    else if(root->data<key)
    {
        search(root->right, key);
    }
}

void Postorder(struct node* root)
{     if (root ==
NULL)
    {
return;
    }
    Postorder(root->left);
Postorder(root->right);
printf("%d ", root->data);
}
```

DS_Writeups_ETC

```
void Inorder(struct node* root)
{     if (root ==
NULL)
    {
return;     }
    Inorder(root->left);
printf("%d ", root->data);
    Inorder(root->right);
}

void Preorder(struct node* root)
{
    if (root == NULL)
    {
return;
    }
    printf("%d ", root->data);     Preorder(root-
>left);
    Preorder(root->right);
}

int main()
{
    int root_data, val, number, i = 0, choice;
printf("Enter the root data: ");
    scanf("%d", &root_data);

    struct node *root = create_node(root_data);

    printf("%d", root->data);


    printf("\nEnter the number of children data in the tree (Max = 10): ");
scanf("%d", &number);

    i = 0;
    while(i<number)
    {
        printf("\nEnter the subsequent data for the children nodes:
");         scanf("%d", &val);         insert(root, val);
        i++;
    }
```

DS_Writeups_ETC

```
    printf("There are mainly 5 operations\n1) Insert\n2) Search\n3) PostOrder
Traversal\n4) PreOrder Traversal\n5) InOrder Traversal\n6) Exit\nEnter the
number corresponding to the serial no. of the operation: ");
scanf("%d", &choice);

    switch (choice)
    {
    case 1: //Insert
    {
        int new;
        printf("\nEnter the new data to be entered in the tree:
");         scanf("%d", &new);         insert(root, new);
        printf("Now by searching through the tree we will ensure if the new value
was inserted properly!");          struct node *n = search(root, new);          if (n-
>data == new)
        {
            printf("\n%d is in the Tree!", new);
        }
else
        {
            printf("\n%d is not present in the Tree!", new);
        }
end();
break;
    }
    case 2: //Search
    {
int key;
        printf("\nEnter the number you want to search in the tree:
");         scanf("%d", &key);         struct node *n = search(root,
key);         if (n->data == key)
        {
            printf("\n%d is in the Tree!", key);
        }
else
        {
            printf("\n%d is not present in the Tree!", key);
        }
end();
break;
    }
    case 3: //PostOrder Traversal
    {
```

DS_Writeups_ETC

```
        printf("\nYou have chosen to perform PostOrder Traversal. Here Traversal
through the tree will be LRD, Left Subtree(L)->Right Subtree(R)-
>Root(D)\n");
        Postorder(root);
        end();
break;
    }
    case 4: //PreOrder Traversal
    {
        printf("\nYou have chosen to perform PostOrder Traversal. Here Traversal
through the tree will be DLR, Root(D)->Left Subtree(L)->Right
Subtree(R)\n");
        Preorder(root);
end();
```

DS_Writeups_ETC

```
        break;
    }
    case 5: //InOrder Traversal
    {
        printf("\nYou have chosen to perform PostOrder Traversal. Here Traversal
through the tree will be LDR, Left Subtree(L)->Root(D)->Right Subtree(R)\n");
        Postorder(root);
        end();
break;
    }
case 6:
    {
end();
break;
    }
default:
    {
        printf("\nInvalid Entry!Please Enter a number between 1 and 6
corresponding to the serial number of the operation as mentioned above!");
end();         break;
    }
}
    return 0;
}
```

CONCLUSION:

In this experiment, we implemented BST using data structures and C programing language and also applied operations like Insert, Search, Recursive Transversal (Preorder, Inorder, Postorder).

REFERENCES:

Seymour Lipschutz, Data Structure with C, Schaum’s Outlines, Tata McGrawHill

E Balgurusamy - Programming in ANSI C, Tata McGraw-Hill (Third Edition)

Yashavant Kanetkar- Let Us C, BPB Publication, 8th Edition.

Continuous Assessment for DS AY 2021-22

RPP (5)	SPO (5)	Total (10)	Signature:
			Assessed By: Mr. V. B. Vaijapurkar

DS_Writeups_ETC

Start date	**Submission date**	**Date:**

22/11/2021	**29/11/2021**	**Roll. No.22108**
***Regularity, Punctuality, performance** ***Submission, Presentation, orals**		

DS_Writeups_ETC