

21/12/2020

## LAB-9 : Binary Search Tree Implementation

- Q. WAP to implement a BST with the following operations:
- Create tree.
  - Traversal (Inorder / Preorder / Postorder)
  - Display.

A. struct node

```
{  
    int key;  
    struct node *left;  
    struct node *right;  
}
```

struct node \*create (int data)

```
{  
    struct node *temp;  
    temp = (struct node *) malloc (sizeof (struct node));  
    temp->key = data;  
    temp->left = temp->right = NULL;  
    return temp;  
}
```

void insert (struct node \*root, struct node \*temp)

```
{  
    if (temp->key < root->key)  
    {  
        if (root->left == NULL)  
            insert (root->left, temp);  
        else  
            root->left = temp;  
    }  
}
```

6/12/2020

```
if (temp → key > root → key)
{
    if (root → right != Null)
        insert (root → right, temp);
    else
        root → right = temp;
}
}
```

```
void inorder (struct node *root)
{
    if (root != Null)
    {
        inorder (root → left);
        printf ("%d", root → key);
        inorder (root → right);
    }
}
```

```
void preorder (struct node *root)
{
    if (root != Null)
    {
        printf ("%d", root → key);
        preorder (root → left);
        preorder (root → right);
    }
}
```

21/12/2020

```
void postorder(struct node *root)
```

```
{
```

```
    if (root != NULL)
```

```
    {
```

```
        postorder (root → left);
```

```
        postorder (root → right);
```

```
        printf ("%d", root → key);
```

```
    }
```

```
}
```

```
int main()
```

```
{
```

```
    char ch;
```

```
    struct node *root = NULL, *temp;
```

```
    do {
```

```
        temp = create (data);
```

```
        if (root == NULL)
```

```
            root = temp;
```

```
        else
```

```
            insert (root, temp);
```

```
        printf ("Do you want to enter more (Y/N)?");
```

```
        getch();
```

```
        scanf ("%c", &ch);
```

```
    } while (ch == 'y' || ch == 'Y');
```

```
    return 0;
```

```
}
```