

LAB -10 : Binary Search Tree

a) Construct & Traversal using inorder, preorder, postorder

Pseudocode :

```

struct tnode {
    int data;
    struct tnode *left, *right;
};
struct tnode *root = NULL;

```

```

struct tnode *createNode(int data) {
    struct tnode *newNode;
    newNode = (struct tnode*) malloc(sizeof(struct tnode));
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return(newNode);
}

```

```

void insertion(struct tnode **node, int data) {
    if (!*node) {
        *node = createNode(data);
    }
    else if (data < (*node)->data) {
        insertion(&(*node)->left, data);
    }
    else if (data > (*node)->data) {
        insertion(&(*node)->right, data);
    }
}

```

```

void inorder(struct tnode *node) {
    if (node) {
        inorder(node->left);
        printf("%d", node->data);
    }
}

```

```

    inorder (node → right);
}
return;
}

```

```

void preorder (struct tnode *node) {
    if (node) {
        printf ("%d", node → data);
        preorder (node → left);
        preorder (node → right);
    }
    return;
}

```

```

void postOrder (struct tnode *node) {
    if (node) {
        postOrder (node → left);
        postOrder (node → right);
        printf ("%d", node → data);
    }
}

```

right return;

```

main() {
    int data, ch;
    while (1) {
        printf ("1. Insertion\n 2. Inorder\n 3. Preorder\n 4. Postorder\n 5. Exit\n Enter your choice\n");
        scanf ("%d", &ch);
        switch (ch) {
            case 1: printf ("Enter data:");
                    scanf ("%d", &data);
                    insertion (&root, data);
                    break;

```



```
case 2: printf("Inorder");
        inorder(root);
        break;
case 3: printf("Preorder");
        preorder(root);
        break;
case 4: printf("Postorder");
        postorder(root);
        break;
case 5: exit(0);
        break;
default: printf("Wrong choice\n");
        break;
}
}
}
return 0
```