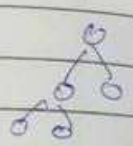
```
Inorder Traversal: 20 30 40 50 60 70 80
Postorder Traversal: 20 40 30 60 80 70 50
Preorder Traversal: 50 30 20 40 70 60 80
```

```
        printf("%d", root->val);
        inorderTraversal(root->right);
    }
}
```

★
```
void preorderTraversal(struct TreeNode* root) {
    if (root != NULL) {
        printf("%d", root->val);          // function to perform preorder traversal
        preorderTraversal(root->left);
        preorderTraversal(root->right);
    }
}
```



★
```
void displayElements(struct TreeNode* root) {
    printf("Inorder traversal: ");
    inorderTraversal(root);               // function to display the elements in the tree
    printf("\n postorder Traversal: ");
    postorderTraversal(root);
    printf("\n Preorder Traversal: ");
    preorderTraversal(root);
    printf("\n");
}
```

Output:

Inorder Traversal : 20 30 40 50 60 70 80

postorder Traversal : 20 40 30 60 80 70 50

preorder Traversal : 50 30 20 40 70 60 80

★
```
void postOrderTraversal(struct TreeNode* root) {
    if (root != NULL) {
        postorderTraversal(root->left);   // function to perform postorder traversal
        postorderTraversal(root->right);
        printf("%d", root->val);
    }
}
```

19.08.24

STUDENTS NAME

19/02/24

SURYA Gold Date Page

BST

```
Struct TreeNode {
    int val;
    Struct TreeNode * left;
    Struct TreeNode * right;
};
```
// definition of binary tree node

```
Struct TreeNode * CreateNode(int val){
    Struct TreeNode *NewNode = (Struct TreeNode)* malloc(
                                sizeof(struct TreeNode));
    NewNode -> val = val;
    NewNode -> left = NULL;
    NewNode -> right = Null;
    return newNode;
}
```
// function to Create a new Node

```
Struct TreeNode * insert(struct TreeNode *root, int val){
    if (root == NULL){
        return CreateNode(val);
    }
    elif (val < root -> val){
        root -> left = insert(root -> left, val);
    }
    else{
        root -> right = Insert (root -> right, val);
    }
    return root;
```
// function to insert a value into the BST

```
Void inorderTraversal(struct TreeNode *root){
    If (root != NULL){
    inorderTraversal(root -> left);
```
// function to perform inorder traversal

SURYA Gold
Date Page

05/02/24

Implimentation of double link list

* Creation of node.

```
Struct node
{
    Struct node *next;
    int data;
    Struct node *prev;
};

Struct node *Create_ll (struct node * Start)
{
    Struct node *new_node, *ptr;
    int num;
    printf("\n Enter -1 to end ");
    printf ("\n Enter the data :");
    Scanf ("%d", &num);
    While (num != -1)
    {
      if (Start == NULL)
      {
        new_node = (struct node *) malloc (size of (Struct node));
        New_node -> Prev = NULL;
        New_node -> data = num;
        new_node -> next = NULL;
        Start = new_node;
      }
      else
      {
        Ptr = Start;
        New_node = (struct node *) malloc (size of (Struct node));
```

SURYA Gold
Date Page

→ Enter your option : 2
20 12 15 45

Enter your option : 3
Enter your option : 3
Enter the data : 14
Enter the value before which the data has to be inserted : 12

→ Enter your option : 2
20 14 12 15 45

→ Enter your option : 4
Enter the value to be deleted :
12

→ Enter your option : 2
20 14 15 45

→ Enter your option : 5

19/02/2
BST
☆ Struct
int
St
St
};
☆ Struct
};
☆ Struct
☆ Void

```
New_node -> data = num;
New_node -> next = NULL;
Start = New_node;
} While (ptr -> next != NULL)
        ptr = ptr -> next;
ptr -> next = new node;
New_node -> prev = ptr;
new_node -> new = NULL;
}
printf ("\n Enter the data:");
Scanf ("%d", &num);
}
return start;

* Struct node * display (struct node * start)
{
    Struct node * ptr;
    ptr = start;
    while (ptr != NULL)
    {
      printf ("\t %d", ptr -> data);
      ptr = ptr -> next;
    }
    return start;
}
```

SURYA Gold
Date Page

```
Struct node *insert_before (struct node * Start)
{
    Struct node * new_node, *Ptr ;
    int num, val;
    Printf ("\n Enter the data :");
    Scanf (" %d", &num);
    printf("\n Enter the value before which the data has to be
                                                    inserted:");
    Scanf ("%d", &val);
    New_node = (struct node*) malloc (sizeof (struct node));
    New_node -> data = num ;
    Ptr = Start;
    While (Ptr -> data != val)
        Ptr = Ptr -> next ;
    New_node -> next = Ptr ;
    New_node -> prev = Ptr -> Prev ;
    Ptr -> Prev -> next = new_node ;
    Ptr -> Prev = new_node ;
    Return Start;
}
```

```
Struct node * delete_selected (Struct node * Start)
{
    Struct node *ptr;
    int val;
    Ptr = Start ;
    printf ("\n Enter the value to be deleted : \n");
    Scanf ("%d", &val);
    While (Ptr -> data != val)
    {
        Ptr = Ptr -> next;
    }
```

```
if (ptr->data == val)
{
    ptr->prev->next = ptr->next;
    ptr->next->prev = ptr->prev;
    free (ptr);
}
else
{
    printf ("Node with %d value doesn't exist \n", val);
}
return start;
}
```

o/p:

```
**MAIN MENU**
1: Create a list
2: Display the list
3: Add a node before a given node
4: Delete a given node
5: EXIT

→ Enter your option : 1
Enter 81 to end
Enter the data : 20
Enter the data : 12
Enter the data : 15
Enter the data : 45
Enter the data : -1

Doubly Linked List Created
```