# Practical – 4

AIM: Implementation and time analysis of Doubly linked list .

```
#include <iostream>
using namespace std;

class Node
{
public:
    int data;
    Node *prev;
    Node *next;

    Node()
    {
        this->data = 0;
        this->prev = NULL;
        this->next = NULL;
    }

    Node(int data)
    {
        this->data = data;
        this->prev = NULL;
        this->next = NULL;
    }
};

void display(Node *head)
{
    if (head == NULL)
    {
        cout << "Empty Linked list!" << endl;
    }
    while (head != NULL)
    {
        cout << head->data << "->";
        head = head->next;
    }
    cout << endl;
}

void insertAtStart(Node *&head)
{
    int data;
    cout << "enter value : ";
    cin >> data;

    Node *newnode = new Node(data);
    if (head == NULL)
    {
        head = newnode;
        return;
    }

    newnode->prev = NULL;
    newnode->next = head;

    head->prev = newnode;
    head = newnode;
}

void insertAtEnd(Node *&head)
{
```

```
    int data;
    cout << "enter value : ";
    cin >> data;

    Node *newnode = new Node(data);

    if (head == NULL)
    {
        head = newnode;
        return;
    }

    Node *temp = head;
    while (temp->next != NULL)
    {
        temp = temp->next;
    }

    temp->next = newnode;
    newnode->prev = temp;
}

void deleteAtStart(Node *&head)
{

    if (head == NULL)
    {
        cout << "empty linked list" << endl;
        return;
    }

    Node *temp = head;

    head = head->next;
    cout << "Deleted successfully" << endl;

    delete temp;
}

void deleteAtEnd(Node *&head)
{

    if (head == NULL)
    {
        cout << "empty linked list" << endl;
        return;
    }

    Node *temp = head;
    Node *prev = NULL;

    while (temp->next != NULL)
    {
        prev = temp;
        temp = temp->next;
    }
    prev->next = NULL;
    cout << "Deleted successfully" << endl;

    delete temp;
}

void insertInOrder(Node *&head)
{

    bool flag_isvalue_fitted = false;

    int data;
```

```
    cout << "enter value : ";
    cin >> data;

    Node *newnode = new Node(data);
    if (head == NULL)
    {
        head = newnode;
        return;
    }

    Node *temp = head;
    int i = 1;

    if (newnode->data < head->data)
    {
        newnode->next = head;
        newnode->prev = NULL;

        head->prev = newnode;

        head = newnode;
        flag_isvalue_fitted = true;
    }
    else
    {

        while (temp->next != NULL)
        {
            cout << " ilteration " << i++ << endl;
            if (temp->data < newnode->data && newnode->data < temp->next->data)
            {
                newnode->prev = temp;
                newnode->next = temp->next;

                temp->next = newnode;

                temp->next->prev = newnode;
                flag_isvalue_fitted = true;
                return;
            }

            temp = temp->next;
        }
    }

    if (flag_isvalue_fitted == false)
    {

        if (temp->next == NULL)
        {
            if (temp->data < newnode->data)
            {
                temp->next = newnode;
                newnode->prev = temp;
            }
        }
    }
}

int main()
{

    Node *head = NULL;
    int choice;

    cout << "1) Show " << endl;
    cout << "2) insert at start " << endl;
```

```
    cout << "3) insert at end " << endl;
    cout << "4) delete at start " << endl;
    cout << "5) delete at end " << endl;
    cout << "6) insert in order (just insert)" << endl;

    do
    {
        cout << "enter choice :";
        cin >> choice;
        switch (choice)
        {
        case 1:
            display(head);
            break;

        case 2:
            insertAtStart(head);
            break;
        case 3:
            insertAtEnd(head);
            break;
        case 4:
            deleteAtStart(head);
            break;
        case 5:
            deleteAtEnd(head);
            break;
        case 6:
            insertInOrder(head);
            break;

        default:
            break;
        }

    } while (choice != 8);
}
```

## Time analysis

| Operation | Time Complexity |
|---|---|
| Insert at Start | O(1) |
| Insert at End | O(n) |
| Delete at Start | O(1) |
| Delete at End | O(n) |
| Insert in Order | O(n) |
| Display | O(n) |

## OUTPUT

```
1) Show
2) insert at start
3) insert at end
4) delete at start
5) delete at end
6) insert in order (just inser
enter choice :1
Empty Linked list!

enter choice :2
enter value : 20
enter choice :2
enter value : 10
enter choice :1
10->20->
enter choice :3
enter value : 30
enter choice :1
10->20->30->
enter choice :6
enter value : 25
enter choice :1
10->20->25->30->
enter choice :4
enter choice :1
20->25->30->
enter choice :5
enter choice :1
20->25->
```