Practical – 3

AIM: Implementation and time analysis of Singly linked list and its applications .

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

class Node
{
public:
    int data;
    Node *next;
    Node()
    {
        data = 0;
    }
    Node(int data)
    {
        this->data = data;
        this->next = NULL;
    }
};

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

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

    newnode->next = head;
    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;
}
```

```
void deleteAtStart(Node *&head)
{
    if (head == NULL)
    {
        cout << "its already empty " << endl;
        return;
    }

    Node *temp = head;
    head = head->next;
    delete temp;
}

void deleteAtEnd(Node *&head)
{
    if (head == NULL)
    {
        cout << "its already empty " << endl;
        return;
    }

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

    prev->next = NULL;
    delete temp;
}

void insertInOrder(Node *&head)
{
    bool flag_isvalue_fitted = false;
    int data;
    cout << "enter Value";
    cin >> data;

    Node *newnode = new Node(data);
    Node *temp = head;

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

    while (temp->next != NULL)
    {
        if (data < head->data)
        {
            flag_isvalue_fitted = true;
            newnode->next = head;
            head = newnode;
        }
        else
        {
            if (data > temp->data && data < temp->next->data)
            {
                flag_isvalue_fitted = true;
                newnode->next = temp->next;
                temp->next = newnode;
            }
        }
        temp = temp->next;
```

```
}
    if (flag_isvalue_fitted == false)
    {
        if (temp->next == NULL)
        {
            temp->next = newnode;
        }
    }
}

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

int main()
{
    Node *head = new Node(100);
    cout << head;

    cout << "1) Display " << 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;

    int choice;

    do
    {
        cout << "enter your 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;
        }

    } while (choice != 10);
}
```

OUTPUT

```
1) Display
2) insert at start
3) insert at end
4) delete at start
5) delete at end
6) insert in order (just insert)
enter your choice :2
enter Value:10
enter your choice :2
enter Value:20
enter your choice :2
enter Value:30
enter your choice :2
enter Value:40
enter your choice :1
40 -> 30 -> 20 -> 10 ->
enter your choice :3
enter Value 100
enter your choice :1
40 -> 30 -> 20 -> 10 -> 100 ->
enter your choice :4
enter your choice :1
30 -> 20 -> 10 -> 100 ->
enter your choice :5
enter your choice :1
30 -> 20 -> 10 ->
enter your choice :6
enter Value 22
enter your choice :1
22 -> 30 -> 20 -> 10 ->
enter your choice :6
enter Value 24
enter your choice :1
22 -> 24 -> 30 -> 20 -> 10 ->
```

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)

Applications

- Dynamic Memory Allocation
- Implementing Stacks and Queues
- Polynomial Representation and Manipulation
- Symbol Tables
- Memory Pools
- Undo Functionality in Software
- Music and Video Playlists
- Expression Evaluation
- Hash Tables
- Graphs Representation (Adjacency Lists)
- Navigation Systems (GPS)
- Undo/Redo Functionality in Editors