

Date: 01.09.2025

## Program – 11

**Aim:** Create a linked list (LL) using nodes having information about students (name, roll, percent marks). Implement:

1. Insertion of a new node at the beginning, at the end, and at any specified position.
2. Deletion of a node from the beginning, from the end, and at any specified position.
3. Reversal of the linked list.

**Theory :** A linked list is a dynamic linear data structure where elements, called nodes, are connected through pointers instead of being stored in contiguous memory.

Each node contains data and a next pointer to the following node.

This allows the list to grow or shrink at runtime without reallocating memory.

Common types include singly linked lists (one-way links), doubly linked lists (links forward and backward), and circular lists (last node points to first).

Basic operations are insertion, deletion, traversal, and reversal.

Insertions and deletions are efficient because only pointers change—no element shifting as in arrays. However, linked lists use extra memory for pointers and allow only sequential access (no direct indexing).

They are well-suited for applications needing frequent insertions or deletions, such as implementing stacks, queues, or dynamic memory management.

### Source Code:

```
#include <iostream>
using namespace std;
class Student
{
public:
    string name;
    int roll;
    float percent;
    Student *next;
};
class LinkedList
{
    Student *head;

public:
    LinkedList() { head = nullptr; }
    void insertBeginning()
    {
        Student *newNode = new Student;
        cout << "Enter name roll percent: ";
        cin >> newNode->name >> newNode->roll >> newNode->percent;
```

```

    newNode->next = head;
    head = newNode;
}
void insertEnd()
{
    Student *newNode = new Student;
    cout << "Enter name roll percent: ";
    cin >> newNode->name >> newNode->roll >> newNode->percent;
    newNode->next = nullptr;
    if (!head)
    {
        head = newNode;
        return;
    }
    Student *temp = head;
    while (temp->next)
        temp = temp->next;
    temp->next = newNode;
}

void insertAtPosition()
{
    int pos;
    cout << "Enter position: ";
    cin >> pos;
    if (pos <= 0)
        return;
    if (pos == 1)
    {
        insertBeginning();
        return;
    }
    Student *newNode = new Student;
    cout << "Enter name roll percent: ";
    cin >> newNode->name >> newNode->roll >> newNode->percent;
    Student *temp = head;
    for (int i = 1; i < pos - 1 && temp; i++)
        temp = temp->next;
    if (!temp)
    {
        cout << "Position out of bounds\n";
        delete newNode;
        return;
    }
    newNode->next = temp->next;
    temp->next = newNode;
}

```

```

void deleteBeginning()
{
    if (!head)
        return;
    Student *temp = head;
    head = head->next;
    delete temp;
}
void deleteEnd()
{
    if (!head)
        return;
    if (!head->next)
    {
        delete head;
        head = nullptr;
        return;
    }
    Student *temp = head;
    while (temp->next && temp->next->next)
        temp = temp->next;
    delete temp->next;
    temp->next = nullptr;
}
void deleteAtPosition()
{
    int pos;
    cout << "Enter position to delete: ";
    cin >> pos;
    if (pos <= 0 || !head)
        return;
    if (pos == 1)
    {
        deleteBeginning();
        return;
    }
    Student *temp = head;
    for (int i = 1; i < pos - 1 && temp; i++)
        temp = temp->next;
    if (!temp || !temp->next)
    {
        cout << "Position out of bounds\n";
        return;
    }
    Student *delNode = temp->next;
    temp->next = delNode->next;
    delete delNode;
}

```

```

    }
    void reverse()
    {
        Student *prev = nullptr;
        Student *current = head;
        Student *next = nullptr;
        while (current)
        {
            next = current->next;
            current->next = prev;
            prev = current;
            current = next;
        }
        head = prev;
    }
    void display()
    {
        Student *temp = head;
        while (temp)
        {
            cout << temp->name << " " << temp->roll << " " << temp->percent << " --> ";
            temp = temp->next;
        }
    }
};

```

```

int main()
{
    LinkedList ll;
    int choice;
    while (true)
    {
        cout << "\n1.Insert Beginning\n2.Insert End\n3.Insert At Position\n4.Delete
Beginning\n5.Delete End\n6.Delete At Position\n7.Reverse\n8.Display\n9.Exit\nEnter
choice: ";
        cin >> choice;
        switch (choice)
        {
            case 1:
                ll.insertBeginning();
                break;
            case 2:
                ll.insertEnd();
                break;
            case 3:
                ll.insertAtPosition();
                break;

```

```
case 4:
    ll.deleteBeginning();
    break;
case 5:
    ll.deleteEnd();
    break;
case 6:
    ll.deleteAtPosition();
    break;
case 7:
    ll.reverse();
    break;
case 8:
    ll.display();
    break;
case 9:
    return 0;
default:
    cout << "Invalid choice\n";
}
}
}
```

## OUTPUT-

```
C:\Users\Akshat\Downloads\c++>cd "c:\Users\Akshat\Downloads\c++\" && g++ 1.cpp
```

```
1.Insert Beginning
2.Insert End
3.Insert At Position
4.Delete Beginning
5.Delete End
6.Delete At Position
7.Reverse
8.Display
9.Exit
Enter choice: 1
Enter name roll percent: Akshat 15 98
```

```
1.Insert Beginning
2.Insert End
3.Insert At Position
4.Delete Beginning
5.Delete End
6.Delete At Position
7.Reverse
8.Display
9.Exit
Enter choice: 2
Enter name roll percent: XYZ 20 88
```

```
1.Insert Beginning
2.Insert End
3.Insert At Position
4.Delete Beginning
5.Delete End
6.Delete At Position
7.Reverse
8.Display
9.Exit
Enter choice: 3
Enter position: 2
Enter name roll percent: Vikas 21 89
```

```
1.Insert Beginning
2.Insert End
3.Insert At Position
4.Delete Beginning
5.Delete End
6.Delete At Position
7.Reverse
8.Display
9.Exit
Enter choice: 5
```

```
1.Insert Beginning
2.Insert End
3.Insert At Position
4.Delete Beginning
5.Delete End
6.Delete At Position
7.Reverse
8.Display
9.Exit
Enter choice: 8
XYZ 20 88 --> Vikas 21 89 -->
1.Insert Beginning
2.Insert End
3.Insert At Position
4.Delete Beginning
5.Delete End
6.Delete At Position
7.Reverse
8.Display
9.Exit
Enter choice: 9
```

```
1.Insert Beginning
2.Insert End
3.Insert At Position
4.Delete Beginning
5.Delete End
6.Delete At Position
7.Reverse
8.Display
9.Exit
Enter choice: 8
Akshat 15 98 --> Vikas 21 89 --> XYZ 20 88 -->
1.Insert Beginning
2.Insert End
3.Insert At Position
4.Delete Beginning
5.Delete End
6.Delete At Position
7.Reverse
8.Display
9.Exit
Enter choice: 7
```

```
1.Insert Beginning
2.Insert End
3.Insert At Position
4.Delete Beginning
5.Delete End
6.Delete At Position
7.Reverse
8.Display
9.Exit
Enter choice: 8
XYZ 20 88 --> Vikas 21 89 --> Akshat 15 98 -->
```

## Program – 12

**AIM:** WAP to delete n nodes after m nodes of the linkedlist.

### CODE:

```
#include <iostream>
using namespace std;
class Node { public:    int data;
Node *next;
Node(int val) : data(val), next(nullptr) {}
}
;
class LinkedList
{
public:
    Node *head;
    LinkedList() : head(nullptr) {}
    void push(int val)
    {
        Node *newNode = new Node(val);
        if (!head)
        {
            head = newNode;
            return;
        }
        Node *temp = head;
        while (temp->next)
        {
            temp = temp->next;
        }
        temp->next = newNode;
    }
    void deleteNAfterM(int m, int n)
    {
        Node *current = head;
        Node *temp;

        while (current)
        {
            for (int i = 1; i < m && current != nullptr; i++)
            {
                current = current->next;
            }
            if (!current)
                return;
            break;
        }
    }
}
```

```

        temp = current->next;
        for (int i = 0; i < n && temp != nullptr; i++)
        {
            temp = temp->next;
        }
        current->next = temp;
        current = temp;
    }
    void printList()
    {
        Node *temp = head;
        while (temp)
        {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
};
int main()
{
    LinkedList ll;
    for (int i = 1; i <= 10; i++)
    {
        ll.push(i);
    }
    cout << "Original list:\n";
    ll.printList();
    int m = 2, n = 3;
    ll.deleteNAfterM(m, n);
    cout << "After deleting " << n << " nodes after every " << m << " nodes:\n";
    ll.printList();
    return 0;
}

```

## OUTPUT-

```

PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE  PORTS

c:\Users\Akshat\Downloads\c++>cd "c:\Users\Akshat\Downloads\c++\" && g++ 1.cpp -o 1 && "c:\Users\Akshat\Downloads\c++\"1
Original list:
1 2 3 4 5 6 7 8 9 10
After deleting 3 nodes after every 2 nodes:
1 2 6 7 8 9 10

c:\Users\Akshat\Downloads\c++>

```



## Program – 13

**AIM:** Write a program to check if a singly linked list is palindrome or not.

### CODE:

```
#include <iostream>
#include <stack>
using namespace std;
class Node { public:
char data;
Node *next;
Node(char val) : data(val), next(nullptr) {}
}
;
class LinkedList
{
public:
Node *head;
LinkedList() : head(nullptr) {}
void push(char val)
{
Node *newNode = new Node(val);
if (!head)
{
head = newNode;
return;
}
Node *temp = head;
while (temp->next)
temp = temp->next;
temp->next = newNode;
}
bool isPalindrome()
{
stack<char> st;
Node *temp = head;
while (temp)
{
st.push(temp->data);
temp = temp->next;
}
temp = head;
while (temp)
{
if (temp->data != st.top())
return false;
}
```

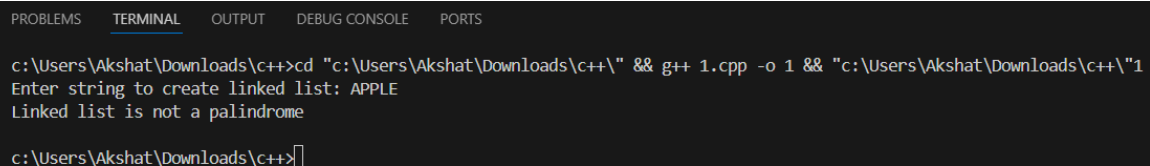
```

        st.pop();
        temp = temp->next;
    }
    return true;
}
};
int main()
{
    LinkedList ll;
    string s;
    cout << "Enter string to create linked list: ";
    cin >> s;
    for (char c : s)
    {
        ll.push(c);
    }
    if (ll.isPalindrome())
        cout << "Linked list is a palindrome\n";
    else
        cout << "Linked list is not a palindrome\n";

    return 0;
}

```

## OUTPUT-



```

PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE  PORTS
c:\Users\Akshat\Downloads\c++>cd "c:\Users\Akshat\Downloads\c++\" && g++ 1.cpp -o 1 && "c:\Users\Akshat\Downloads\c++\1
Enter string to create linked list: APPLE
Linked list is not a palindrome

c:\Users\Akshat\Downloads\c++>

```

## Program – 14

**AIM:** WAP to remove the loop in linked list.

### CODE:

```
#include <iostream>
#include <bits/stdc++.h>
using namespace std;

struct Node
{
    int data;
    Node *next;
    Node(int d) : data(d), next(nullptr) {}
};

void removeLoop(Node *head)
{
    if (!head || !head->next)
        return;

    Node *slow = head, *fast = head;
    bool loopFound = false;

    while (fast && fast->next)
    {
        slow = slow->next;
        fast = fast->next->next;
        if (slow == fast)
        {
            loopFound = true;
            break;
        }
    }

    if (!loopFound)
        return;
    slow = head;
    if (slow != fast)
    {
        while (slow->next != fast->next)
        {
            slow = slow->next;
```

```

        fast = fast->next;
    }
    fast->next = nullptr;
}
else
{

    while (fast->next != slow)
        fast = fast->next;
    fast->next = nullptr;
}
}

void printList(Node *head)
{
    Node *temp = head;
    while (temp)
    {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << "\n";
}

int main()
{
    Node *head = new Node(1);
    head->next = new Node(2);
    head->next->next = new Node(3);
    head->next->next->next = new Node(4);
    head->next->next->next->next = new Node(5);

    head->next->next->next->next->next = head->next->next;

    removeLoop(head);

    cout << "Linked List after removing loop:\n";
    printList(head);

    return 0;
}

```

## OUTPUT:

```
PROBLEMS  TERMINAL  OUTPUT  DEBUG CONSOLE  PORTS

c:\Users\Akshat\Downloads\c++>cd "c:\Users\Akshat\Downloads\c++\" && g++ 1.cpp -o 1 && "c:\Users\Akshat\Downloads\c++\1
Linked List after removing loop:
1 2 3 4 5

c:\Users\Akshat\Downloads\c++>
```