PIMPRI CHINCHWAD EDUCATION TRUST's.

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

(An Autonomous Institute)

Class : SY BTech **Acad. Yr. 2025-26** **Semester : I**

Name of the student: Om Jitendra Khalane **PRN : 124B1B040**

Department: Computer Engineering **Division : A**

Course Name : Data Structures and Laboratory **Course Code: BCE23PC02**

Completion Date : 04/08/2025

Assignment No. 04

Problem Statement:

Design a music playlist system using a linked list where:
- Songs can be added to the beginning/end
- Songs can be deleted
- Next and previous songs can be navigated

Source Code :

https://github.com/omkhalane/DSAL-SY-PCCOE/blob/main/lab_assignments/assignment04.cpp

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

class playlist
{
public:
    string data;
    playlist *next;
    playlist()
    {
```

```
            next = nullptr;
        }
        playlist(string song_name)
        {
            data = song_name;
            next = NULL;
        }
    };
    class ll
    {
    public:
        playlist *head;
        playlist *tail;
        playlist *currentsong;
        ll()
        {
            head = NULL;
            tail = NULL;
            currentsong = NULL;
        }

        void insert_song_end(string NameAtEnd)
        {
            playlist *newsong = new playlist(NameAtEnd);
            if (head == NULL)
            {
                head = newsong;
                tail = newsong;
                currentsong = newsong;
                return;
```

```
        }

        tail->next = newsong;
        tail = newsong;
    }
    void insert_song_beg(string NameAtBeg)
    {
        playlist *newsong = new playlist(NameAtBeg);
        if (head == NULL)
        {
            head = newsong;
            tail = newsong;
            currentsong = newsong;
            return;
        }

        newsong->next = head;
        head = newsong;
    }

    void display_songs()
    {
        if (head == NULL)
        {
            cout << "The list of song is empty!!" << endl;
            return;
        }
        else
        {
            playlist *temp = head;
```

```
            while (temp != NULL)
            {
                cout << temp->data << " -> ";
                temp = temp->next;
            }
            cout << "NULL " << endl;
        }
    }

    void deletesong(string ToDelete)
    {
        if (head == nullptr)
        {
            cout << "The list of songs is empty!!" << endl;
            return;
        }

        if (head->data == ToDelete)
        {
            playlist *toDeleteNode = head;
            head = head->next;
            if (currentsong == toDeleteNode)
                currentsong = head;
            if (head == nullptr)
                tail = nullptr;
            delete toDeleteNode;
            cout << "Song deleted successfully!" << endl;
            return;
        }
```

```
            playlist *temp = head;
            while (temp->next != nullptr && temp->next->data != ToDelete)
            {
                temp = temp->next;
            }

            if (temp->next == nullptr)
            {
                cout << "Song not found in playlist!" << endl;
                return;
            }

            playlist *toDeleteNode = temp->next;
            temp->next = toDeleteNode->next;

            if (toDeleteNode == tail)
            {
                tail = temp;
            }
            if (currentsong == toDeleteNode)
            {
                currentsong = temp;
            }

            delete toDeleteNode;
            cout << "Song deleted successfully!" << endl;
        }

        void NextSong()
        {
```

```
        if (currentsong == nullptr)
        {
            cout << "Playlist is empty!" << endl;
            return;
        }
        if (currentsong->next != nullptr)
        {
            currentsong = currentsong->next;
            cout << "Now playing: " << currentsong->data << endl;
        }
        else
        {
            cout << "currently on first song" << endl;
        }
    }

    void PreviousSong()
    {
        if (currentsong == nullptr)
        {
            cout << "Playlist is empty!" << endl;
            return;
        }
        if (currentsong == head)
        {
            cout << "last song reached" << endl;
            return;
        }

        playlist *temp = head;
```

```
            while (temp->next != currentsong)
            {
                temp = temp->next;
            }
            currentsong = temp;
            cout << "Now playing: " << currentsong->data << endl;
        }
    };
    int main()
    {
        int choice;
        ll l1;
        while (true)
        {
            string sname;
                    cout << "\n=============== WELCOME TO YOUR MUSIC PLAYER
===============\n";
            cout << "1: Add song at the beginning\n";
            cout << "2: Add song at the end\n";
            cout << "3: Display playlist\n";
            cout << "4: Next song\n";
            cout << "5: Previous song\n";
            cout << "6: Delete song\n";
            cout << "7: Exit\n";
                                                                cout    <<
"=========================================================\n";
            cout << "Enter choice: ";
            cin >> choice;
            cin.ignore();

            switch (choice)
```

```
{
case 1:
    cout << "Enter the song to add at the beginning: ";
    getline(cin, sname);
    l1.insert_song_beg(sname);
    break;
case 2:
    cout << "Enter the song to add at the end: ";
    getline(cin, sname);
    l1.insert_song_end(sname);
    break;
case 3:
    cout << "\n============= PLAYLIST =============\n";
    l1.display_songs();
    break;
case 4:
    l1.NextSong();
    break;
case 5:
    l1.PreviousSong();
    break;
case 6:
    cout << "Enter song name to delete: ";
    getline(cin, sname);
    l1.deletesong(sname);
    break;
case 7:
    cout << "Exiting playlist. Goodbye!\n";
    return 0;
default:
```

```
                cout << "Invalid choice! Try again.\n";
            }
        }
    }
```

Screen Shot of Output :

```
============== WELCOME TO YOUR MUSIC PLAYER ==============
1: Add song at the beginning
2: Add song at the end
3: Display playlist
4: Next song
5: Previous song
6: Delete song
7: Exit
============================================================
Enter choice: 3

============= PLAYLIST =============
The list of song is empty!!

============== WELCOME TO YOUR MUSIC PLAYER ==============
1: Add song at the beginning
2: Add song at the end
3: Display playlist
4: Next song
5: Previous song
6: Delete song
7: Exit
============================================================
Enter choice: 1
Enter the song to add at the beginning: DemoSong01

============== WELCOME TO YOUR MUSIC PLAYER ==============
1: Add song at the beginning
2: Add song at the end
3: Display playlist
4: Next song
5: Previous song
6: Delete song
7: Exit
============================================================
Enter choice: 2
Enter the song to add at the end: DemoSong02

============== WELCOME TO YOUR MUSIC PLAYER ==============
1: Add song at the beginning
2: Add song at the end
3: Display playlist
4: Next song
5: Previous song
6: Delete song
7: Exit
============================================================
Enter choice: 3

============= PLAYLIST =============
DemoSong01 -> DemoSong02 -> NULL
```

```
============== WELCOME TO YOUR MUSIC PLAYER ==============
1: Add song at the beginning
2: Add song at the end
3: Display playlist
4: Next song
5: Previous song
6: Delete song
7: Exit
============================================================
Enter choice: 1
Enter the song to add at the beginning: DemoSong03

============== WELCOME TO YOUR MUSIC PLAYER ==============
1: Add song at the beginning
2: Add song at the end
3: Display playlist
4: Next song
5: Previous song
6: Delete song
7: Exit
============================================================
Enter choice: 3

============= PLAYLIST =============
DemoSong03 -> DemoSong01 -> DemoSong02 -> NULL

============== WELCOME TO YOUR MUSIC PLAYER ==============
1: Add song at the beginning
2: Add song at the end
3: Display playlist
4: Next song
5: Previous song
6: Delete song
7: Exit
============================================================
Enter choice: 4
Now playing: DemoSong02

============== WELCOME TO YOUR MUSIC PLAYER ==============
1: Add song at the beginning
2: Add song at the end
3: Display playlist
4: Next song
5: Previous song
6: Delete song
7: Exit
============================================================
Enter choice: 5
Now playing: DemoSong01
```

```
=============== WELCOME TO YOUR MUSIC PLAYER ===============
1: Add song at the beginning
2: Add song at the end
3: Display playlist
4: Next song
5: Previous song
6: Delete song
7: Exit
============================================================
Enter choice: 6
Enter song name to delete: DemoSong01
Song deleted successfully!

=============== WELCOME TO YOUR MUSIC PLAYER ===============
1: Add song at the beginning
2: Add song at the end
3: Display playlist
4: Next song
5: Previous song
6: Delete song
7: Exit
============================================================
Enter choice: 3

============= PLAYLIST =============
DemoSong03 -> DemoSong02 -> NULL

=============== WELCOME TO YOUR MUSIC PLAYER ===============
1: Add song at the beginning
2: Add song at the end
3: Display playlist
4: Next song
5: Previous song
6: Delete song
7: Exit
============================================================
Enter choice: 6
Enter song name to delete: DemoSong02
Song deleted successfully!

=============== WELCOME TO YOUR MUSIC PLAYER ===============
1: Add song at the beginning
2: Add song at the end
3: Display playlist
4: Next song
5: Previous song
6: Delete song
7: Exit
============================================================
Enter choice: 7
Exiting playlist. Goodbye!
[1] + Done                       "/usr/bin/gdb" --interpreter=mi
```

Conclusion:

The implemented program demonstrates the use of a **singly linked list** to create a music playlist management system. The application supports **insertion** of songs at both the beginning and end, **deletion** of songs, and **navigation** through the playlist using "Next" and "Previous" functionality.

- **Time Complexity:**
    - **Insertion at Beginning:** O(1)O(1)O(1) — Performed in constant time by adjusting the head pointer.
    - **Insertion at End:** O(1)O(1)O(1) — Achieved in constant time using a tail pointer.
    - **Deletion:** O(n)O(n)O(n) — Requires traversal to locate the song before removal.
    - **Next Song Navigation:** O(1)O(1)O(1) — Moves directly to the next node.
    - **Previous Song Navigation:** O(n)O(n)O(n) — Requires traversal from the head to locate the previous node.
    - **Display Playlist:** O(n)O(n)O(n) — Visits each song once for output.
- **Space Complexity:**
    - Each song is stored as a node containing a string and a pointer, resulting in O(n)O(n)O(n) space usage, where nnn is the number of songs in the playlist.
- **Observations:**
    - The linked list structure allows dynamic addition and deletion of songs without shifting elements, unlike arrays.
    - The **tail pointer** ensures efficient insertion at the end.
    - Navigation to the previous song is less efficient in a singly linked list due to the need for traversal; using a **doubly linked list** could optimize this to O(1)O(1)O(1).

This design effectively demonstrates linked list operations in a real-world application context, providing an interactive and flexible playlist management system.