PIMPRI CHINCHWAD EDUCATION TRUST's.

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

(An Autonomous Institute)

S.Y. B. TECH **Year:** 2024 – 25 **Semester:** I

Name: Sonawane Prachi Mahendra. **PRN:** 124B2B018

Department : Computer Engineering **Division:** B

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

Date: 31/08/24

Assignment – 3

- **Aim:**

  Consider the playlist in a music player. Implement a playlist feature in music player application using singly linked list. Each song in the playlist is represented as a node in the linked list. Each node contains information about the song (such as title or artist or duration, etc.). The playlist should allow users to:

  1. Add songs
  2. Remove songs
  3. Display the entire playlist
  4. Play specific songs

- **Source Code:**

```
#include <iostream>

#include <string>


class Song

{

private:
```

```
    std::string title;
    std::string artist; int
    duration; Song* next;
public:
    Song(std::string title1, std::string artist1, int d)
        : title(title1), artist(artist1), duration(d), next(nullptr) {}
    std::string getTitle() const { return title; } std::string
    getArtist() const { return artist; } int getDuration()
    const { return duration; }
    Song* getNext() const { return next; }
    void setNext(Song* nextSong) { next = nextSong; }
};
class Playlist
{
private:
    Song* head;
public:
    Playlist() : head(nullptr) {}
    void addSong(std::string title, std::string artist, int duration)
    {
        Song* newSong = new Song(title, artist, duration); if
        (!head)
        {
```

```
        head = newSong;
    }
    else
    {
        Song* temp = head; while
        (temp->getNext())
        {
            temp = temp->getNext();
        }
        temp->setNext(newSong);
    }
    std::cout << "Added: " << title << " by " << artist << std::endl;
}
void removeSong(const std::string& title)
{
    if (!head)
    {
        std::cout << "Playlist is empty." << std::endl; return;
    }
    Song* temp = head; Song* prev =
    nullptr;
    while (temp && temp->getTitle() != title)
    {
        prev = temp;
```

```
        temp = temp->getNext();
    }
    if (!temp)
    {
        std::cout << "Song not found: " << title << std::endl; return;
    }
    if (prev)
    {
        prev->setNext(temp->getNext());
    }
    else
    {
        head = temp->getNext(); // Remove head
    }
    delete temp;
    std::cout << "Removed: " << title << std::endl;
}
void displayPlaylist()
{
    if (!head)
    {
        std::cout << "Playlist is empty." << std::endl; return;
    }
    Song* temp = head;
```

```
    std::cout << "Playlist:" << std::endl; while
    (temp)
    {
        std::cout << "Title: " << temp->getTitle()
                << ", Artist: " << temp->getArtist()
        << ", Duration: " << temp->getDuration() << " seconds" << std::endl; temp =
        temp->getNext();
    }
}
void playSong(const std::string title)
{
    Song* temp = head; while
    (temp)
    {
        if (temp->getTitle() == title)
        {
         std::endl;
        }
          std::cout << "Playing: " << temp->getTitle()
                  << " by " << temp->getArtist()
<< " [Duration: " << temp->getDuration() << " seconds]" << return;

        temp = temp->getNext();
    }
```

```
        std::cout << "Song not found: " << title << std::endl;
    }
};
    int main() { Playlist
    playlist; int choice;
    std::string title, artist; int
    duration;

    do {
std::cout << "\n1. Add Song\n2. Remove Song\n3. Display Playlist\n4. Play Song\n5. Exit\n";
        std::cout << "Enter your choice: "; std::cin
        >> choice;

            switch (choice) { case
            1:
                std::cout << "Enter song title: ";
                std::cin.ignore(); std::getline(std::cin,
                title); std::cout << "Enter artist name: ";
                std::getline(std::cin, artist);
                std::cout << "Enter duration (in seconds): "; std::cin
                >> duration;

                playlist.addSong(title, artist, duration); break;
            case 2:
```

```
            std::cout << "Enter song title to remove: ";
            std::cin.ignore();
            std::getline(std::cin, title);
            playlist.removeSong(title); break;
        case 3:
            playlist.displayPlaylist(); break;
        case 4:
            std::cout << "Enter song title to play: ";
            std::cin.ignore();
            std::getline(std::cin, title);
            playlist.playSong(title); break;
        case 5:
            std::cout << "Exiting..." << std::endl; break;
        default:
            std::cout << "Invalid choice. Please try again." << std::endl;
    }
} while (choice != 5);
return 0;
```

- **Screen shots of Output:**

1.

```
Output
/tmp/FFR9Z3JI3D.o

1. Add Song
2. Remove Song
3. Display Playlist
4. Play Song
5. Exit
Enter your choice: 1
Enter song title: Barish
Enter artist name: Arijit
Enter duration (in seconds): 3
Added: Barish by Arijit

1. Add Song
2. Remove Song
3. Display Playlist
4. Play Song
5. Exit
Enter your choice: 1
Enter song title: Lover
Enter artist name: taylor swift
Enter duration (in seconds): 4
Added: Lover by taylor swift
```

Output

```
1. Add Song
2. Remove Song
3. Display Playlist
4. Play Song
5. Exit
Enter your choice: 3
Playlist:
Title: Barish, Artist: Arijit, Duration: 3 seconds
Title: Lover, Artist: taylor swift, Duration: 4 seconds

1. Add Song
2. Remove Song
3. Display Playlist
4. Play Song
5. Exit
Enter your choice: 2
Enter song title to remove: Barish
Removed: Barish

1. Add Song
2. Remove Song
3. Display Playlist
4. Play Song
5. Exit
Enter your choice: 3
Playlist:
Title: Lover, Artist: taylor swift, Duration: 4 seconds
```

```
1. Add Song
2. Remove Song
3. Display Playlist
4. Play Song
5. Exit
Enter your choice: 5
Exiting...
```

- **Conclusion:**

  Hence, we studied about singly linked list and its operations like insertion, deletion, traversing, etc.