

NAME : PREM SAH
PRN : 123B1B234
DIV : D

ASSIGNMENT NO.3

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:

- a. Add songs
- b. Remove songs
- c. Display the entire playlist
- d. Play specific songs

CODE :

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

// Node structure for each song in the playlist
class SongNode {
public:
    string title;
    string artist;
    int duration; // duration in seconds
    SongNode* next;

    SongNode(string t, string a, int d) : title(t), artist(a), duration(d), next(nullptr) {}
};

// Playlist class using singly linked list
class Playlist {
private:
    SongNode* head;

public:
    Playlist() : head(nullptr) {}

    // Add song to the end of the playlist
    void addSong(string title, string artist, int duration) {
        SongNode* newSong = new SongNode(title, artist, duration);
        if (!head) {
```

```

    head = newSong;
} else {
    SongNode* temp = head;
    while (temp->next != nullptr) {
        temp = temp->next;
    }
    temp->next = newSong;
}
cout << "Song added: " << title << " by " << artist << endl;
}

```

// Remove song by title

```

void removeSong(string title) {
    if (!head) {
        cout << "Playlist is empty!" << endl;
        return;
    }

```

```

    SongNode* temp = head;
    SongNode* prev = nullptr;

```

// If the song to remove is the head

```

if (temp != nullptr && temp->title == title) {
    head = temp->next;
    delete temp;
    cout << "Song removed: " << title << endl;
    return;
}

```

// Search for the song in the playlist

```

while (temp != nullptr && temp->title != title) {
    prev = temp;
    temp = temp->next;
}

```

// If song not found

```

if (temp == nullptr) {
    cout << "Song not found!" << endl;
    return;
}

```

// Unlink the node and free memory

```

prev->next = temp->next;
delete temp;

```

```

    cout << "Song removed: " << title << endl;
}

// Display the entire playlist
void displayPlaylist() {
    if (!head) {
        cout << "The playlist is empty!" << endl;
        return;
    }

    SongNode* temp = head;
    int count = 1;
    while (temp != nullptr) {
        cout << count++ << ". " << temp->title << " by " << temp->artist
            << " [" << temp->duration / 60 << " min " << temp->duration % 60 << " sec]" <<
endl;
        temp = temp->next;
    }
}

// Play a specific song by title
void playSong(string title) {
    SongNode* temp = head;
    while (temp != nullptr) {
        if (temp->title == title) {
            cout << "Now playing: " << temp->title << " by " << temp->artist << endl;
            return;
        }
        temp = temp->next;
    }
    cout << "Song not found in the playlist!" << endl;
}

};

int main() {
    Playlist myPlaylist;

    // Adding songs to the playlist
    myPlaylist.addSong("Song One", "Artist A", 210);
    myPlaylist.addSong("Song Two", "Artist B", 240);
    myPlaylist.addSong("Song Three", "Artist C", 180);

    // Display playlist
    cout << "\nCurrent Playlist:" << endl;

```

```
myPlaylist.displayPlaylist();

// Playing a specific song
cout << "\nPlaying a specific song:" << endl;
myPlaylist.playSong("Song Two");

// Removing a song
cout << "\nRemoving a song:" << endl;
myPlaylist.removeSong("Song One");

// Display updated playlist
cout << "\nUpdated Playlist:" << endl;
myPlaylist.displayPlaylist();

return 0;
}
```

Output :

Song added: Song One by Artist A
Song added: Song Two by Artist B
Song added: Song Three by Artist C

Current Playlist:

- 1. Song One by Artist A [3 min 30 sec]**
- 2. Song Two by Artist B [4 min 0 sec]**
- 3. Song Three by Artist C [3 min 0 sec]**

Playing a specific song:

Now playing: Song Two by Artist B

Removing a song:

Song removed: Song One

Updated Playlist:

- 1. Song Two by Artist B [4 min 0 sec]**
- 2. Song Three by Artist C [3 min 0 sec]**

main.cpp

```
1 #include <iostream>
2 #include <string>
3 using namespace std;
4
5 // Node structure for each song in the playlist
6 class SongNode {
7 public:
8     string title;
9     string artist;
10    int duration; // duration in seconds
11    SongNode* next;
12
13    SongNode(string t, string a, int d) : title(t), artist(a), duration(d), next(nullptr) {}
14 };
15
16 // Playlist class using singly linked list
17 class Playlist {
18 private:
19     SongNode* head;
20
21 public:
22     Playlist() : head(nullptr) {}
23
24     // Add song to the end of the playlist
25     void addSong(string title, string artist, int duration) {
26         SongNode* newSong = new SongNode(title, artist, duration);
27         if (!head) {
28             head = newSong;
29         } else {
30             SongNode* temp = head;
31             while (temp->next != nullptr) {
32                 temp = temp->next;
33             }
34             temp->next = newSong;
35         }
36         cout << "Song added: " << title << " by " << artist << endl;
37     }
38
39     // Remove song by title
40     void removeSong(string title) {
41         if (!head) {
42             cout << "Playlist is empty!" << endl;
43             return;
44         }
45
46         SongNode* temp = head;
47         SongNode* prev = nullptr;
```

main.cpp

```
48
49 // If the song to remove is the head
50 if (temp != nullptr && temp->title == title) {
51     head = temp->next;
52     delete temp;
53     cout << "Song removed: " << title << endl;
54     return;
55 }
56
57 // Search for the song in the playlist
58 while (temp != nullptr && temp->title != title) {
59     prev = temp;
60     temp = temp->next;
61 }
62
63 // If song not found
64 if (temp == nullptr) {
65     cout << "Song not found!" << endl;
66     return;
67 }
68
69 // Unlink the node and free memory
70 prev->next = temp->next;
71 delete temp;
72 cout << "Song removed: " << title << endl;
73 }
74
75 // Display the entire playlist
76 void displayPlaylist() {
77     if (!head) {
78         cout << "The playlist is empty!" << endl;
79         return;
80     }
81
82     SongNode* temp = head;
83     int count = 1;
84     while (temp != nullptr) {
85         cout << count++ << ". " << temp->title << " by " << temp->artist
86             << " [" << temp->duration / 60 << " min " << temp->duration % 60 << " sec]" << endl;
87         temp = temp->next;
88     }
89 }
90
91 // Play a specific song by title
92 void playSong(string title) {
93     SongNode* temp = head;
94     while (temp != nullptr) {
```

```

95-         if (temp->title == title) {
96-             cout << "Now playing: " << temp->title << " by " << temp->artist << endl;
97-             return;
98-         }
99-         temp = temp->next;
100-     }
101-     cout << "Song not found in the playlist!" << endl;
102- }
103- };
104-
105- int main() {
106-     Playlist myPlaylist;
107-
108-     // Adding songs to the playlist
109-     myPlaylist.addSong("Song One", "Artist A", 210);
110-     myPlaylist.addSong("Song Two", "Artist B", 240);
111-     myPlaylist.addSong("Song Three", "Artist C", 180);
112-
113-     // Display playlist
114-     cout << "\nCurrent Playlist:" << endl;
115-     myPlaylist.displayPlaylist();
116-
117-     // Playing a specific song
118-     cout << "\nPlaying a specific song:" << endl;
119-     myPlaylist.playSong("Song Two");
120-
121-     // Removing a song
122-     cout << "\nRemoving a song:" << endl;
123-     myPlaylist.removeSong("Song One");
124-
125-     // Display updated playlist
126-     cout << "\nUpdated Playlist:" << endl;
127-     myPlaylist.displayPlaylist();
128-
129-     return 0;
130- }
131-

```

Output

/tmp/NuGZtHEg93.o

Song added: Song One by Artist A
 Song added: Song Two by Artist B
 Song added: Song Three by Artist C

Current Playlist:

1. Song One by Artist A [3 min 30 sec]
 2. Song Two by Artist B [4 min 0 sec]
 3. Song Three by Artist C [3 min 0 sec]

Playing a specific song:

Now playing: Song Two by Artist B

Removing a song:

Song removed: Song One

Updated Playlist:

1. Song Two by Artist B [4 min 0 sec]
 2. Song Three by Artist C [3 min 0 sec]

=== Code Execution Successful ===