Project Report:

Music Playlist Manager

1. Introduction

The Music Playlist Manager is a C++ program designed to manage a playlist of songs using a doubly linked list data structure. It supports basic operations such as adding songs, deleting songs, printing the playlist, and playing the current song. This project provides a foundation for managing a music playlist in an efficient manner using linked lists.

2. Objectives

- To design a program that can manage a playlist of songs.
- To implement basic playlist operations such as insertion, deletion, and playback.
- To utilize the doubly linked list data structure for efficient management of the playlist.
- To ensure proper memory management and resource deallocation.

3. System Design

3.1 Node Structure

Each song in the playlist is represented as a node in a doubly linked list. The `Node` structure contains:

- A `title` to store the name of the song.
- A pointer to the previous node (`prev`).
- A pointer to the next node (`next`).

3.2 MusicPlaylistManager Class

The `MusicPlaylistManager` class manages the playlist operations and maintains pointers to the head and tail of the linked list. It includes methods to insert, delete, print, and play songs.

4. Implementation

4.1 Node Structure

```
struct Node {
    string title;
    Node* prev;
    Node* next;
};
```

This structure defines a node in the doubly linked list, containing the song's title and pointers to the previous and next nodes.

4.2 MusicPlaylistManager Methods

```
class MusicPlaylistManager {
private:
    Node* head;
    Node* tail;

public:
    MusicPlaylistManager() {
        head = nullptr;
        tail = nullptr;
    }

    void insert(const string& title) {
        Node* newNode = new Node();
        newNode->title = title;
        newNode->prev = tail;
        newNode->next = nullptr;

        if (tail != nullptr) {
            tail->next = newNode;
        } else {
            head = newNode;
        }

        tail = newNode;
    }

    void deleteSong(const string& title) {
        Node* curr = head;
        while (curr != nullptr) {
            if (curr->title == title) {
                if (curr->prev != nullptr) {
                    curr->prev->next = curr->next;
                } else {
                    head = curr->next;
                }

                if (curr->next != nullptr) {
                    curr->next->prev = curr->prev;
                } else {
                    tail = curr->prev;
                }

                delete curr;
                return;
            }
            curr = curr->next;
        }
    }

    void play() {
        if (head != nullptr) {
```

```
            cout << "Playing: " << head->title << endl;
        } else {
            cout << "Playlist is empty." << endl;
        }
    }

    void print() {
        Node* curr = head;
        while (curr != nullptr) {
            cout << curr->title << endl;
            curr = curr->next;
        }
    }

    ~MusicPlaylistManager() {
        Node* curr = head;
        while (curr != nullptr) {
            Node* next = curr->next;
            delete curr;
            curr = next;
        }
    }
};
```

This class includes methods to manage the playlist:

- `insert(const string& title)`: Adds a new song to the end of the playlist.
- `deleteSong(const string& title)`: Deletes a song from the playlist by title.
- `play()`: Plays the current song at the head of the playlist.
- `print()`: Prints the entire playlist.
- Destructor: Ensures proper memory deallocation by deleting all nodes in the linked list.

4.3 Main Function

```
int main() {
    MusicPlaylistManager playlist;

    playlist.insert("Song 1");
    playlist.insert("Song 2");
    playlist.insert("Song 3");

    cout << "Playlist:" << endl;
    playlist.print();

    playlist.deleteSong("Song 2");

    cout << "Playlist after deleting 'Song 2':" << endl;
    playlist.print();

    playlist.play();

    return 0;
}
```

The main function demonstrates the usage of the `MusicPlaylistManager` class:

- Creates an instance of `MusicPlaylistManager`.
- Inserts three songs into the playlist.
- Prints the initial playlist.
- Deletes "Song 2" from the playlist.
- Prints the updated playlist.
- Plays the current song at the head of the playlist.

5. Testing and Results

The program was tested with a series of operations including insertion, deletion, printing, and playing songs. The following output was observed:

```
Playlist:
Song 1
Song 2
Song 3
Playlist after deleting 'Song 2':
Song 1
Song 3
Playing: Song 1
```

The output confirms that the program correctly handles the insertion and deletion of songs, and properly prints and plays the playlist.

6. Conclusion

The Music Playlist Manager successfully implements a doubly linked list to manage a playlist of songs. It allows for efficient insertion and deletion of songs, and provides functionalities to print and play the playlist. The project demonstrates the effective use of data structures and memory management in C++.

7. Future Enhancements

Future improvements to the Music Playlist Manager could include:

- Implementing a shuffle function to randomly reorder the playlist.
- Adding functionality to play the next and previous songs.
- Enhancing the user interface to allow more interactive commands.
- Adding persistent storage to save and load playlists from a file.
- Implementing additional error handling and validation for user inputs.

This project report provides a comprehensive overview of the Music Playlist Manager, including its design, implementation, and results. The report highlights the effectiveness of using a doubly linked list for managing a music playlist and suggests potential future enhancements to improve functionality and user experience.

Output: