Kristu Jayanti College
AUTONOMOUS Bengaluru
Reaccredited A++ Grade by NAAC | Affiliated to Bengaluru North University

NAME: ALFIN JOSE

REG.NO.: 24MCAA06

COURSE TITLE: DATA STRUCTURE AND ALGORITHM

TOPIC: MUSIC PLAYER MANAGER USING LINKED LIST

SUBMITTED TO: TOMIN JOSEPH

NO. OF PAGES: 10

SUBMITTED ON: 20/10/24

ABOUT THE SYSTEM

This C program implements a simple **Music Playlist Manager** using **Linked Lists**. It allows the user to add, delete, shuffle, and display songs in a playlist. The playlist is represented as a singly linked list where each node is a song.

Key Components:

- Song Structure:

  - Contains a `**title**` (string) and a `**next**` pointer to the next song.

  - `**head**` is a global pointer that points to the first song in the list. If `**head**` is `**NULL**`, the playlist is empty.

Main Features:

1. Add Song (`insertSong`): Adds a song to the end of the playlist.

2. Delete Song (`deleteSong`): Removes a song by title from the playlist. If not found, it prints a message.

3. Shuffle Playlist (`shuffleSongs`): Randomly shuffles the playlist by storing songs in an array, shuffling, and then reconstructing the linked list.

4. Display Playlist (`displayPlaylist`): Lists all the songs in the current order or indicates if the playlist is empty.

User Menu:

- The program provides a menu where the user can:

  1. Add a song

  2. Delete a song

  3. Shuffle the playlist

  4. Display the playlist

  5. Exit the program

Memory Management:

- The program uses `malloc()` to dynamically allocate memory for new songs and `free()` to delete them.

This system allows users to manage and shuffle a dynamic collection of songs through a simple console interface.

DATA STRUCTURE USED

The data structure used in this music playlist system is a **Singly Linked List**, where each node represents a song in the playlist. The nodes are dynamically allocated using `malloc()` and contain two elements: the song's title (stored as a character array) and a pointer to the next song. The playlist begins with a `head` pointer, which points to the first song in the list, and if `head` is `NULL`, the playlist is considered empty.

In this structure, songs are added to the end of the list by traversing from the head to the last node, where the new song is inserted. Deletion is performed by searching for the target song and updating the pointers to bypass and free the memory associated with the deleted node.

The key benefit of using a linked list is its dynamic nature, allowing efficient memory usage without the need for resizing, as would be required in an array. Additionally, insertion and deletion operations can be performed in constant time once the correct position is found, unlike arrays where shifting elements would be necessary. This flexibility makes the linked list well-suited for managing the dynamic nature of a playlist.

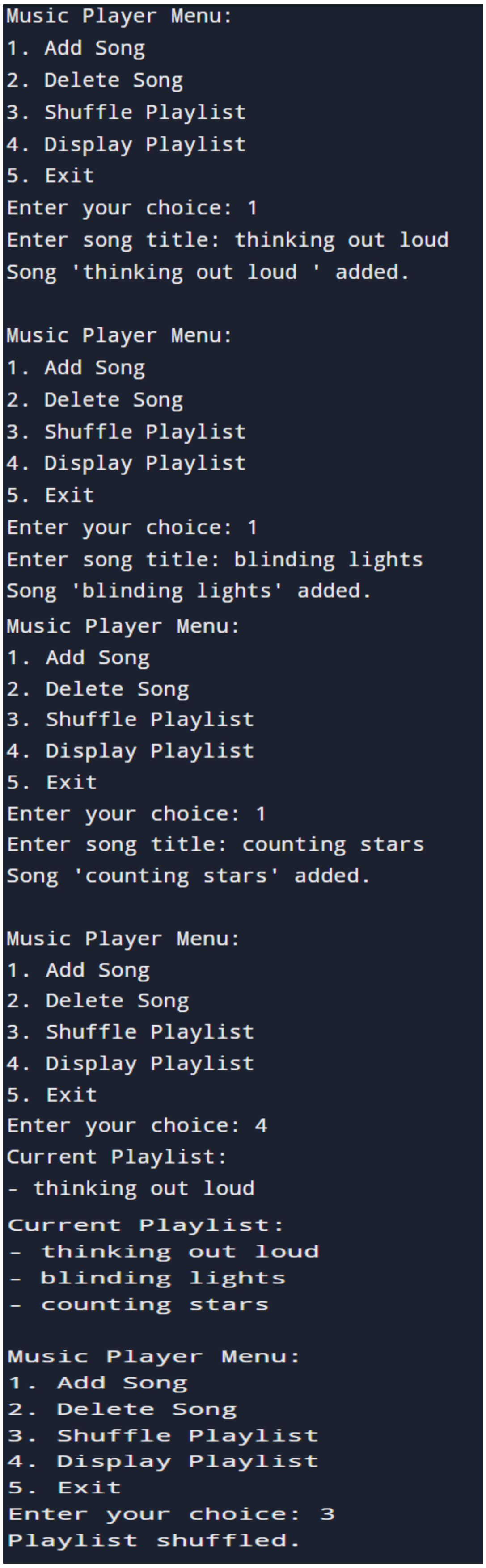
For the shuffle feature, the linked list is temporarily converted into an array of pointers for easy reordering, which allows the songs to be randomly shuffled. Once shuffled, the array is converted back into the linked list structure. This system effectively manages a playlist, offering efficient memory usage and quick modifications to the song order or content.

CODE

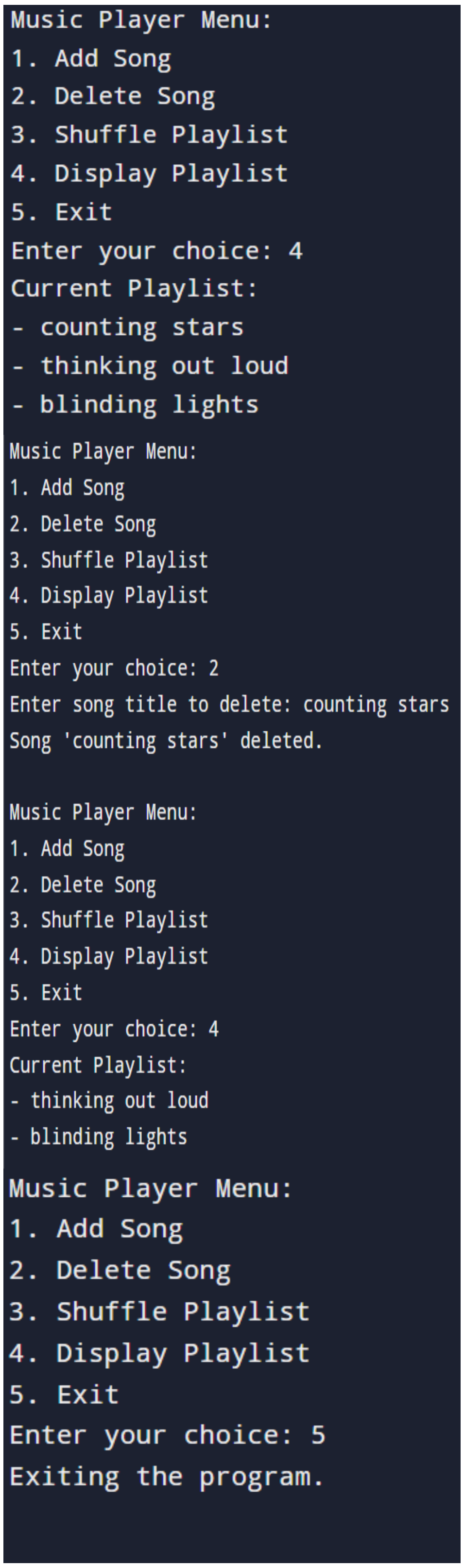
```c
#include <stdio.h>

#include <stdlib.h>

#include <string.h>


typedef struct Song {

   char title[50];

   struct Song* next;

} Song;


Song* head = NULL;


// Function to create a new song

Song* createSong(char* title) {

   Song* newSong = (Song*)malloc(sizeof(Song));

   strcpy(newSong->title, title);

   newSong->next = NULL;

   return newSong;

}


// Function to insert a song at the end

void insertSong(char* title) {

   Song* newSong = createSong(title);

   if (head == NULL) {

      head = newSong;

   } else {

      Song* temp = head;
```

```
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = newSong;
    }
    printf("Song '%s' added.\n", title);
}

// Function to delete a song by title
void deleteSong(char* title) {
    if (head == NULL) {
        printf("The playlist is empty.\n");
        return;
    }
    Song* temp = head;
    Song* prev = NULL;

    while (temp != NULL && strcmp(temp->title, title) != 0) {
        prev = temp;
        temp = temp->next;
    }

    if (temp == NULL) {
        printf("Song '%s' not found.\n", title);
        return;
    }

    if (prev == NULL) {
        head = temp->next;
    } else {
        prev->next = temp->next;
    }
    free(temp);
    printf("Song '%s' deleted.\n", title);
}

// Function to shuffle the playlist
void shuffleSongs() {
    if (head == NULL) {
        printf("The playlist is empty.\n");
        return;
    }

    int count = 0;
    Song* temp = head;

    // Count the number of songs
    while (temp != NULL) {
        count++;
        temp = temp->next;
    }

    // Create an array to store song pointers
    Song** songArray = (Song*)malloc(count * sizeof(Song));
    temp = head;
    for (int i = 0; i < count; i++) {
        songArray[i] = temp;
        temp = temp->next;
    }

    // Shuffle the array
    for (int i = 0; i < count; i++) {
        int j = rand() % count;
        Song* tempSong = songArray[i];
        songArray[i] = songArray[j];
        songArray[j] = tempSong;
    }

    // Reconstruct the linked list from the shuffled array
    head = songArray[0];
    temp = head;
    for (int i = 1; i < count; i++) {
        temp->next = songArray[i];
        temp = temp->next;
    }
    temp->next = NULL;

    free(songArray);
    printf("Playlist shuffled.\n");
}
// Function to display the playlist
void displayPlaylist() {
    if (head == NULL) {
        printf("The playlist is empty.\n");
        return;
    }
    Song* temp = head;
    printf("Current Playlist:\n");
    while (temp != NULL) {
        printf("- %s\n", temp->title);
        temp = temp->next;
    }
}

int main() {
    int choice;
    char title[50];

    do {
        printf("\nMusic Player Menu:\n");
        printf("1. Add Song\n");
        printf("2. Delete Song\n");
        printf("3. Shuffle Playlist\n");
        printf("4. Display Playlist\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        getchar();  // To consume the newline character
        switch (choice) {
            case 1:
                printf("Enter song title: ");
                fgets(title, sizeof(title), stdin);
                title[strcspn(title, "\n")] = 0;  // Remove newline character
                insertSong(title);
                break;
            case 2:
                printf("Enter song title to delete: ");
                fgets(title, sizeof(title), stdin);
                title[strcspn(title, "\n")] = 0;  // Remove newline character
                deleteSong(title);
                break;
            case 3:
                shuffleSongs();
                break;
            case 4:
                displayPlaylist();
                break;
            case 5:
                printf("Exiting the program.\n");
                break;
            default:
                printf("Invalid choice! Please try again.\n");
        }
    } while (choice != 5);
    return 0;
}
```

OUTPUT



```
Music Player Menu:
1. Add Song
2. Delete Song
3. Shuffle Playlist
4. Display Playlist
5. Exit
Enter your choice: 1
Enter song title: thinking out loud
Song 'thinking out loud ' added.

Music Player Menu:
1. Add Song
2. Delete Song
3. Shuffle Playlist
4. Display Playlist
5. Exit
Enter your choice: 1
Enter song title: blinding lights
Song 'blinding lights' added.
Music Player Menu:
1. Add Song
2. Delete Song
3. Shuffle Playlist
4. Display Playlist
5. Exit
Enter your choice: 1
Enter song title: counting stars
Song 'counting stars' added.

Music Player Menu:
1. Add Song
2. Delete Song
3. Shuffle Playlist
4. Display Playlist
5. Exit
Enter your choice: 4
Current Playlist:
- thinking out loud
Current Playlist:
- thinking out loud
- blinding lights
- counting stars

Music Player Menu:
1. Add Song
2. Delete Song
3. Shuffle Playlist
4. Display Playlist
5. Exit
Enter your choice: 3
Playlist shuffled.
```



```
Music Player Menu:
1. Add Song
2. Delete Song
3. Shuffle Playlist
4. Display Playlist
5. Exit
Enter your choice: 4
Current Playlist:
- counting stars
- thinking out loud
- blinding lights
Music Player Menu:
1. Add Song
2. Delete Song
3. Shuffle Playlist
4. Display Playlist
5. Exit
Enter your choice: 2
Enter song title to delete: counting stars
Song 'counting stars' deleted.

Music Player Menu:
1. Add Song
2. Delete Song
3. Shuffle Playlist
4. Display Playlist
5. Exit
Enter your choice: 4
Current Playlist:
- thinking out loud
- blinding lights
Music Player Menu:
1. Add Song
2. Delete Song
3. Shuffle Playlist
4. Display Playlist
5. Exit
Enter your choice: 5
Exiting the program.
```