Assignment
Spring 2025

Course Title: Data Structures

COURSE CODE : CSE 1301

SECTION : 1

Submitted To:
Prof. Dr. Muhamad Golam Kibria
University of Liberal Arts Bangladesh

Submitted By:
Md. Tazminur Rahman Tanim
Id- 242014124
University of Liberal Arts Bangladesh

Date: 20/03/2025

Department of CSE

University of Liberal Arts Bangladesh (ULAB)

Problem Statement :

Write a program that allows users to create a music playlist consisting of n songs and then display the list of the songs. The program should prompt user to enter the name of each song and add it to the playlist in a sequential order. The playlist should be designed to repeat once it reaches the end of the list.

Discussion on the Program :

The goal of this program is to create a music playlist in C using a **circular linked list**, allowing users to input a specified number of songs and store them in sequential order. Once the playlist reaches the last song, it loops back to the beginning, ensuring continuous playback. The program prompts the user to enter the number of songs (`n`) and then input each song's name. After storing all songs, the program displays the complete playlist in order.

Key Features of the Program:

1. **User Input for Playlist Size:**
    - The program prompts the user to specify how many songs they want in the playlist.
2. **Sequential Song Addition:**
    - Users input song names one by one, and they are stored in a circular linked list in the order entered.
3. **Circular Playlist Implementation:**
    - The playlist loops back to the first song after reaching the last, simulating continuous playback.
4. **Playlist Display:**
    - After all songs are added, the program displays the full list of songs in the entered order.
5. **Continuous Playback Mechanism:**
    - Users can skip to the next song, and the playlist automatically repeats after the last song.
6. **Simple User Interface:**
    - A user-friendly interface ensures easy interaction and seamless music navigation.

Advantages of Using a Circular Linked List Instead of an Array:

A **circular linked list** is ideal for implementing a music playlist because:

- It allows seamless looping by pointing the last node back to the first.
- It is dynamic, meaning memory allocation is done as needed, avoiding the fixed size limitation of arrays.
- It provides efficient insertion and deletion of songs without requiring shifts like in arrays.

- It enables smooth song transitions without complex index management.

In contrast, arrays:

- Have a fixed size, leading to potential wasted memory or insufficient space.
- Require shifting elements when adding or removing songs, making them inefficient for dynamic playlists.
- Do not naturally support looping without additional logic.

Functions in the Program:

1. **createnode()**
    - Creates a new node dynamically for storing a song.
    - Uses malloc to allocate memory.
    - Assigns the given song name to the node and initializes its next pointer to NULL.
2. **insertsong()**
    - Adds a new song to the circular linked list.
    - Calls createnode(name) to create a new song node.
    - If the playlist is empty (*head == NULL), the new node becomes the first song and points to itself.
    - Otherwise, it finds the last node and updates its next pointer to point to the new song, maintaining the circular structure.
3. **displayplaylist()**
    - Prints all songs in the playlist in the order they were entered.
4. **playSongs()**
    - Simulates song playback by moving through the circular linked list.
    - Waits for user input (e.g., pressing n) to proceed to the next song.
    - Once the last song is reached, it loops back to the first, ensuring infinite playback.
5. **freePlaylist()**
    - Frees allocated memory to prevent memory leaks when the program terminates.

Implement tools : VS Code

Screenshot of input :

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct node {
    char name[100];
    struct node *next;
};

struct node *createnode(char name[]) {
    struct node* newnode = (struct node*)malloc(sizeof(struct node));
    if (newnode == NULL) {
        printf("Memory allocation failed!\n");
        exit(1);
    }
    strcpy(newnode->name, name);
    newnode->next = NULL;
    return newnode;
}

void insertsong(struct node **head, char name[]) {
    struct node *newnode = createnode(name);
    if (*head == NULL) {
        *head = newnode;
        newnode->next = *head;
    } else {
        struct node* temp = *head;
        while (temp->next != *head) {
            temp = temp->next;
        }
        temp->next = newnode;
        newnode->next = *head;
    }
}
```

```c
void displayplaylist(struct node *head) {
    if (head == NULL) {
        printf("Playlist is empty!\n");
        return;
    }
    struct node *temp = head;
    printf("\nYour Playlist:\n");
    do {
        printf("%s\n", temp->name);
        temp = temp->next;
    } while (temp != head);
}

void playSongs(struct node *head) {
    if (head == NULL) {
        printf("No songs in the playlist!\n");
        return;
    }

    struct node *temp = head;
    char choice;

    while (1) {
        printf("\nNow playing: %s\n", temp->name);
        printf("Press 'n' for next song: ");
        scanf(" %c", &choice);
        if (choice == 'n') {
            temp = temp->next;
        }
    }
}
```

```c
int main() {
    struct node *playlist = NULL;
    int n;
    char songname[100];


    printf("Enter the number of songs: ");
    scanf("%d", &n);
    getchar();

    for (int i = 0; i < n; i++) {
        printf("Enter song %d name: ", i + 1);
        fgets(songname, sizeof(songname), stdin);
        songname[strcspn(songname, "\n")] = 0;
        insertsong(&playlist, songname);
    }

    displayplaylist(playlist);

    playSongs(playlist);

    return 0;
}
```

Screenshot of Output :