

# Rajalakshmi Engineering College

Name: Shremathi K  
Email: 240701504@rajalakshmi.edu.in  
Roll no:  
Phone: 8870649491  
Branch: REC  
Department: I CSE FE  
Batch: 2028  
Degree: B.E - CSE

Scan to verify results



## NeoColab\_REC\_CS23231\_DATA STRUCTURES

### REC\_DS using C\_Week 2\_COD\_Question 1

Attempt : 1  
Total Mark : 10  
Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

Your task is to create a program to manage a playlist of items. Each item is represented as a character, and you need to implement the following operations on the playlist.

Here are the main functionalities of the program:

Insert Item: The program should allow users to add items to the front and end of the playlist. Items are represented as characters. Display Playlist: The program should display the playlist containing the items that were added.

To implement this program, a doubly linked list data structure should be used, where each node contains an item character.

#### ***Input Format***

The input consists of a sequence of space-separated characters, representing the items to be inserted into the doubly linked list.

The input is terminated by entering - (hyphen).

### ***Output Format***

The first line of output prints "Forward Playlist: " followed by the linked list after inserting the items at the end.

The second line prints "Backward Playlist: " followed by the linked list after inserting the items at the front.

Refer to the sample output for formatting specifications.

### ***Sample Test Case***

Input: a b c -

Output: Forward Playlist: a b c

Backward Playlist: c b a

### ***Answer***

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
struct Node {  
    char item;  
    struct Node* next;  
    struct Node* prev;  
};
```

```
void insertAtEnd(struct Node** head, char item) {  
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));  
    newNode->item = item;  
    newNode->next = NULL;  
    newNode->prev = NULL;
```

```
    // If the list is empty, the new node becomes both the head and the tail  
    if (*head == NULL) {  
        *head = newNode;  
    } else {
```

```

    struct Node* temp = *head;
    // Traverse to the last node
    while (temp->next != NULL) {
        temp = temp->next;
    }
    // Insert the new node at the end
    temp->next = newNode;
    newNode->prev = temp;
}
}

// Function to display the playlist from head to tail (forward order)
void displayForward(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%c ", temp->item);
        temp = temp->next;
    }
    printf("\n");
}

// Function to display the playlist from tail to head (backward order)
void displayBackward(struct Node* tail) {
    struct Node* temp = tail;
    while (temp != NULL) {
        printf("%c ", temp->item);
        temp = temp->prev;
    }
    printf("\n");
}

// Function to free the memory allocated for the playlist
void freePlaylist(struct Node* head) {
    struct Node* temp;
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }
}

int main() {
    struct Node* playlist = NULL;

```

```
char item;

while (1) {
    scanf(" %c", &item);
    if (item == '-') {
        break;
    }
    insertAtEnd(&playlist, item);
}

struct Node* tail = playlist;
while (tail->next != NULL) {
    tail = tail->next;
}

printf("Forward Playlist: ");
displayForward(playlist);

printf("Backward Playlist: ");
displayBackward(tail);

freePlaylist(playlist);

return 0;
}
```

**Status :** Correct

**Marks :** 10/10