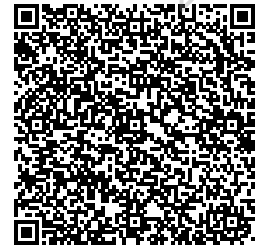


Rajalakshmi Engineering College

Name: srihari krishnakumar
Email: 241501093@rajalakshmi.edu.in
Roll no: 241501093
Phone: 7200067930
Branch: REC
Department: I AI & ML FA
Batch: 2028
Degree: B.E - AI & ML

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;  
};
```

```
// Function declarations
```

```
struct Node* createNode(char item);
```

```
void insertAtEnd(struct Node** head, struct Node** tail, char item);
```

```
void insertFront(struct Node** head, struct Node** tail, char item);
```

```
void displayForward(struct Node* head);
```

```
void displayBackward(struct Node* tail);
```

```
void freePlaylist(struct Node* head);
```

```
// Create a new node
```

```
struct Node* createNode(char item) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    newNode->item = item;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}
```

// Insert at the end

```
void insertAtEnd(struct Node** head, char item) {
    struct Node* newNode = createNode(item);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
```

```
    struct Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->prev = temp;
}
```

// Insert at the front

```
void insertFront(struct Node** head, struct Node** tail, char item) {
    struct Node* newNode = createNode(item);
    if (*head == NULL) {
        *head = *tail = newNode;
    } else {
        newNode->next = *head;
        (*head)->prev = newNode;
        *head = newNode;
    }
}
```

```
// Display from head to tail
void displayForward(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%c ", temp->item);
        temp = temp->next;
    }
    printf("\n");
}
```

```
// Display from tail to head
void displayBackward(struct Node* tail) {
    struct Node* temp = tail;
    while (temp != NULL) {
        printf("%c ", temp->item);
        temp = temp->prev;
    }
    printf("\n");
}
```

```
// Free memory of 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