Rajalakshmi Engineering College

Name: JEENESHWAAR .S
Email: 241501073@rajalakshmi.edu.in
Roll no: 241501073
Phone: 9884283976
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 : 0

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;
};

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

// Define the structure for a doubly linked list node
typedef struct Node {
   char item;
   struct Node* next;
   struct Node* prev;
} Node;
```

```
// Function to create a new node
Node* createNode(char item) {
   Node* newNode = (Node*)malloc(sizeof(Node));
   newNode->item = item;
   newNode->next = NULL;
   newNode->prev = NULL;
   return newNode;
}

// Function to insert an item at the end of the doubly linked list
void insertAtEnd(Node** head, char item) {
   Node* newNode = createNode(item);
   if (*head == NULL) {
      *head = newNode;
      return;
   }
   Node* temp = *head;
   while (temp->next != NULL) {
      temp = temp->next;
   }
   temp->next = newNode;
   newNode->prev = temp;
}

// Function to insert an item at the front of the doubly linked list
void insertAtFront(Node** head, char item) {
   Node* newNode = createNode(item);
   if (*head == NULL) {
      *head = newNode;
      return;
   }
   newNode->next = *head;
   (*head)->prev = newNode;
   *head = newNode;
}

// Function to display the playlist in forward order
void displayForward(Node* head) {
   Node* temp = head;
   while (temp != NULL) {
      printf("%c ", temp->item);
```

```
        temp = temp->next;
    }
}

// Function to display the playlist in backward order
void displayBackward(Node* head) {
    Node* temp = head;
    if (temp == NULL) return;

    // Move to the end of the list
    while (temp->next != NULL) {
        temp = temp->next;
    }

    // Print in reverse order
    while (temp != NULL) {
        printf("%c ", temp->item);
        temp = temp->prev;
    }
}

int main() {
    Node* head = NULL;
    char input[100];

    // Read input until a hyphen is encountered
    while (1) {
        scanf("%s", input);
        if (strcmp(input, "-") == 0) {
            break;
        }
        // Insert at the end for forward playlist
        insertAtEnd(&head, input[0]);
        // Insert at the front for backward playlist
        insertAtFront(&head, input[0]);
    }

    // Display the playlists
    printf("Forward Playlist: ");
    displayForward(head);
    printf("\nBackward Playlist: ");
    displayBackward(head);
    printf("\n");

    // Free the allocated memory (not shown here for brevity)
    // Ideally, you should free the linked list nodes to avoid memory leaks

    return 0;
}

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 : Wrong   ***Marks : 0/10***