Rajalakshmi Engineering College

Name: Subhalakshmi M
Email: 240701539@rajalakshmi.edu.in
Roll no: 240701539
Phone: 6379032776
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 **list, char e) {
   struct Node*newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->item = e;
   newNode->next = NULL;

   if (*list == NULL) {
      newNode->prev = NULL;
      *list = newNode;
   } else {
        struct Node *position = *list;
        while (position->next != NULL) {
            position = position->next;
        }
        position->next = newNode;
        newNode->prev = position;
    }
}

void displayForward(struct Node *list) {
    struct Node *position = list;
    while (position != NULL) { // Traverse from head to the end
        printf("%c ", position->item);
        position = position->next;
    }
    printf("\n");
}

void displayBackward(struct Node *list) {
    struct Node *position = list;
    if (position == NULL) {
        printf( "\n");
        return;
    }

    while (position->next != NULL) {
        position = position->next;
    }

    while (position != NULL) {
        printf("%c ", position->item);
        position = position->prev;
    }
    printf("\n");
}

void freePlaylist(struct Node *list) {
    struct Node *position = list;
    struct Node *temp;
    while (position != NULL) {
        temp = position;
        position = position->next;
        free(temp); // Free each node
    }
}

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***