Rajalakshmi Engineering College

Name: Jhanani shree
Email: 240701215@rajalakshmi.edu.in
Roll no: 240701215
Phone: 7373333511
Branch: REC
Department: I CSE AH
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;
};
```

```
#include <iostream>
#include <sstream>
#include <string>
using namespace std;

struct Node {
    char item;
    Node* next;
    Node* prev;
    Node(char x) : item(x), next(nullptr), prev(nullptr) {}
```

```
};

void insert_at_end(Node** head, Node** tail, char item) {
    Node* new_node = new Node(item);

    if (*head == nullptr) {
        *head = *tail = new_node;
    } else {
        (*tail)->next = new_node;
        new_node->prev = *tail;
        *tail = new_node;
    }
}


void insert_at_front(Node** head, Node** tail, char item) {
    Node* new_node = new Node(item);

    if (*head == nullptr) {
        *head = *tail = new_node;
    } else {
        new_node->next = *head;
        (*head)->prev = new_node;
        *head = new_node;
    }
}

void display_forward(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->item << " ";
        temp = temp->next;
    }
    cout << endl;
}

void display_backward(Node* tail) {
    Node* temp = tail;
    while (temp != nullptr) {
        cout << temp->item << " ";
        temp = temp->prev;
    }
```

```
    cout << endl;
}

int main() {
    Node* head = nullptr;
    Node* tail = nullptr;
    string input;

    getline(cin, input);

    stringstream ss(input);
    char item;

    while (ss >> item) {
        if (item == '-') break;
        insert_at_end(&head, &tail, item);
    }

    cout << "Forward Playlist: ";
    display_forward(head);

    cout << "Backward Playlist: ";
    display_backward(tail);

    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 : Correct ***Marks : 10/10***