

Rajalakshmi Engineering College

Name: tharunika R

Email: 241801296@rajalakshmi.edu.in

Roll no: 241801296

Phone: 6369646218

Branch: REC

Department: I AI & DS FD

Batch: 2028

Degree: B.E - AI & DS

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;  
};
```

```
// You are using GCC
```

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
// Define the structure of the doubly linked list node
```

```
struct Node {  
    char data;  
    Node* next;  
    Node* prev;
```

```
// Constructor to create a new node
Node(char item) {
    data = item;
    next = nullptr;
    prev = nullptr;
}
};
```

```
// Function to insert a node at the end of the list
void insertEnd(Node*& head, Node*& tail, char item) {
    Node* newNode = new Node(item);
    if (head == nullptr) {
        head = newNode;
        tail = newNode;
    } else {
        tail->next = newNode;
        newNode->prev = tail;
        tail = newNode;
    }
}
```

```
// Function to insert a node at the front of the list
void insertFront(Node*& head, Node*& tail, char item) {
    Node* newNode = new Node(item);
    if (head == nullptr) {
        head = newNode;
        tail = newNode;
    } else {
        newNode->next = head;
        head->prev = newNode;
        head = newNode;
    }
}
```

```
// Function to display the playlist from head to tail
void displayForward(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
}
```

```

    cout << endl;
}

// Function to display the playlist from tail to head
void displayBackward(Node* tail) {
    Node* temp = tail;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->prev;
    }
    cout << endl;
}

```

```

int main() {
    Node* head = nullptr; // Head pointer for the playlist
    Node* tail = nullptr; // Tail pointer for the playlist

    string input;

    // Read the input until we encounter a hyphen '-'
    while (cin >> input && input != "-") {
        // Insert each character at the end of the playlist
        insertEnd(head, tail, input[0]);
    }

    // Display the playlist forward
    cout << "Forward Playlist: ";
    displayForward(head);

    // Display the playlist backward
    cout << "Backward Playlist: ";
    displayBackward(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