Rajalakshmi Engineering College

Name: Harjeet V
Email: 240801111@rajalakshmi.edu.in
Roll no: 240801111
Phone: 8148388668
Branch: REC
Department: I ECE FB
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ravi is developing a student registration system for a college. To efficiently store and manage the student IDs, he decides to implement a doubly linked list where each node represents a student's ID.

In this system, each student's ID is stored sequentially, and the system needs to display all registered student IDs in the order they were entered.

Implement a program that creates a doubly linked list, inserts student IDs, and displays them in the same order.

Input Format

The first line contains an integer N the number of student IDs.

The second line contains N space-separated integers representing the student IDs.

Output Format

The output should display the single line containing N space-separated integers representing the student IDs stored in the doubly linked list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50
Output: 10 20 30 40 50

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
    int studentID;
    struct Node* next;
    struct Node* prev;
} Node;

typedef struct {
    Node* head;
    Node* tail;
    int size;
} DoublyLinkedList;

DoublyLinkedList* createList() {
    DoublyLinkedList* list = (DoublyLinkedList*)malloc(sizeof(DoublyLinkedList));
    if (list == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    list->head = NULL;
    list->tail = NULL;
```

```
    list->size = 0;
    return list;
}

void insertAtEnd(DoublyLinkedList* list, int studentID) {

    Node* newNode = (Node*)malloc(sizeof(Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    newNode->studentID = studentID;
    newNode->next = NULL;

    if (list->head == NULL) {
        newNode->prev = NULL;
        list->head = newNode;
        list->tail = newNode;
    } else {

        newNode->prev = list->tail;
        list->tail->next = newNode;
        list->tail = newNode;
    }

    list->size++;
}

void displayList(DoublyLinkedList* list) {
    Node* current = list->head;

    while (current != NULL) {
        printf("%d ", current->studentID);
        current = current->next;
    }
    printf("\n");
}

void freeList(DoublyLinkedList* list) {
    Node* current = list->head;
    Node* next;
```

```
    while (current != NULL) {
        next = current->next;
        free(current);
        current = next;
    }

    free(list);
}

int main() {
    int N, studentID;

    DoublyLinkedList* studentList = createList();

    scanf("%d", &N);

    for (int i = 0; i < N; i++) {
        scanf("%d", &studentID);
        insertAtEnd(studentList, studentID);
    }

    displayList(studentList);

    freeList(studentList);

    return 0;
}
```

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