Rajalakshmi Engineering College

Name: vinnarasan j
Email: 241501245@rajalakshmi.edu.in
Roll no: 241501245
Phone: 7305999373
Branch: REC
Department: I AI & ML FC
Batch: 2028
Degree: B.E - AI & ML

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>

// Structure for a node in the doubly linked list
struct Node {
    int id;
    struct Node* prev;
    struct Node* next;
};

// Function to insert a new node at the end of the list
struct Node* insertEnd(struct Node* head, int student_id) {
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    if (new_node == NULL) {
        printf("Memory allocation failed\n");
        exit(EXIT_FAILURE);
    }
    new_node->id = student_id;
    new_node->next = NULL;

    if (head == NULL) {
        new_node->prev = NULL;
        return new_node;
```

```
    }

    struct Node* current = head;
    while (current->next != NULL) {
        current = current->next;
    }
    current->next = new_node;
    new_node->prev = current;
    return head;
}

// Function to display the linked list
void displayList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->id);
        current = current->next;
    }
    printf("\n");
}

// Function to free the memory allocated for the linked list
void freeList(struct Node* head) {
    struct Node* current = head;
    struct Node* next;
    while (current != NULL) {
        next = current->next;
        free(current);
        current = next;
    }
}

int main() {
    int n, student_id;
    struct Node* head = NULL;

    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%d", &student_id);
        head = insertEnd(head, student_id);
    }
```

```
    displayList(head);
    freeList(head);
    return 0;
}
```

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