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 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Moniksha, a chess coach organizing a tournament, needs a program to manage participant IDs efficiently. The program maintains a doubly linked list of IDs and offers two functions: Append to add IDs as students register, and Print Maximum ID to identify the highest ID for administrative tasks.

This tool streamlines tournament organization, allowing Moniksha to focus on coaching her students effectively.

Input Format

The first line consists of an integer n, representing the number of participant IDs to be added.

The second line consists of n space-separated integers representing the participant IDs.

Output Format

The output displays a single integer, representing the maximum participant ID.

If the list is empty, the output prints "Empty list!".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
163 137 155
Output: 163

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 append a new ID to the end of the list
struct Node* appendID(struct Node* head, int new_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 = new_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 find the maximum ID in the list
int findMaxID(struct Node* head) {
    if (head == NULL) {
        printf("Empty list!\n");
        return -1; // Or some other indicator of an empty list
    }

    int max_id = head->id;
    struct Node* current = head->next;
    while (current != NULL) {
        if (current->id > max_id) {
            max_id = current->id;
        }
        current = current->next;
    }
    return max_id;
}

// 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, id;
    struct Node* head = NULL;
```

```
    scanf("%d", &n);

    if (n > 0) {
        for (int i = 0; i < n; i++) {
            scanf("%d", &id);
            head = appendID(head, id);
        }

        int max_id = findMaxID(head);
        if (max_id != -1) {
            printf("%d\n", max_id);
        }
    } else {
        printf("Empty list!\n");
    }

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

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