Rajalakshmi Engineering College

Name: tharunika R
Email: 241801296@rajalakshmi.edu.in
Roll no: 241801296
Phone: 6369646218
Branch: REC
Department: l 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 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.

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

```c
#include <stdio.h>
#include <stdlib.h>

// Node structure for doubly linked list
struct Node {
    int id;
    struct Node* prev;
    struct Node* next;
};

// Create a new node
struct Node* createNode(int id) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->id = id;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

// Append node to the end
void append(struct Node** head, struct Node** tail, int id) {
    struct Node* newNode = createNode(id);
    if (*head == NULL) {
        *head = *tail = newNode;
    } else {
```

```c
            (*tail)->next = newNode;
            newNode->prev = *tail;
            *tail = newNode;
        }
    }

    // Find and return the maximum ID
    int findMaxID(struct Node* head) {
        if (head == NULL) return -1; // Indicates empty list
        int max = head->id;
        struct Node* current = head->next;
        while (current != NULL) {
            if (current->id > max) {
                max = current->id;
            }
            current = current->next;
        }
        return max;
    }

    int main() {
        int n;
        scanf("%d", &n);

        struct Node* head = NULL;
        struct Node* tail = NULL;

        if (n == 0) {
            printf("Empty list!\n");
            return 0;
        }

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

        int maxID = findMaxID(head);
        if (maxID == -1) {
            printf("Empty list!\n");
        } else {
```

```
        printf("%d\n", maxID);
    }

    return 0;
}
```

Status : Correct   *Marks : 10/10*