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 1_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Imagine you are tasked with developing a simple GPA management system using a singly linked list. The system allows users to input student GPA values, insertion should happen at the front of the linked list, delete record by position, and display the updated list of student GPAs.

Input Format

The first line of input contains an integer n, representing the number of students.

The next n lines contain a single floating-point value representing the GPA of each student.

The last line contains an integer position, indicating the position at which a student record should be deleted. Position starts from 1.

Output Format

After deleting the data in the given position, display the output in the format "GPA: " followed by the GPA value, rounded off to one decimal place.

Refer to the sample output for formatting specifications.

Sample Test Case

```
Input: 4
3.8
3.2
3.5
4.1
2
Output: GPA: 4.1
GPA: 3.2
GPA: 3.8
```

Answer

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

// Structure for a node in the linked list
struct Node {
    float gpa;
    struct Node* next;
};

// Function to insert a new node at the front of the linked list
struct Node* insertFront(struct Node* head, float new_gpa) {
    struct Node* new_node = (struct Node*)malloc(sizeof(struct Node));
    if (new_node == NULL) {
        printf("Memory allocation failed\n");
        exit(EXIT_FAILURE);
    }
    new_node->gpa = new_gpa;
    new_node->next = head;
    return new_node;
}

// Function to delete a node at a given position
struct Node* deleteAtPosition(struct Node* head, int position) {
    if (head == NULL) {
        return NULL;
    }
    if (position == 1) {
        struct Node* temp = head;
        head = head->next;
        free(temp);
        return head;
    }
    struct Node* current = head;
    struct Node* previous = NULL;
    int count = 1;
    while (current != NULL && count < position) {
        previous = current;
        current = current->next;
        count++;
    }
    if (current == NULL) {
        printf("Position out of bounds\n");
        return head;
    }
    previous->next = current->next;
    free(current);
    return head;
}

// Function to display the linked list
void displayList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("GPA: %.1f\n", roundf(current->gpa * 10) / 10);
        current = current->next;
    }
}

// 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, position;
    float gpa;
    struct Node* head = NULL;

    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%f", &gpa);
        head = insertFront(head, gpa);
    }
    scanf("%d", &position);

    head = deleteAtPosition(head, position);

    displayList(head);

    freeList(head);

    return 0;
}
```

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