# Rajalakshmi Engineering College

Name: Ravi Sankar
Email: 240701424@rajalakshmi.edu.in
Roll no: 240701424
Phone: 8122932671
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

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***

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

typedef struct Node {
    float gpa;
    struct Node* next;
} Node;

// Insert GPA at the front
void insertFront(Node** head, float gpa) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->gpa = gpa;
    newNode->next = *head;
    *head = newNode;
}

// Delete node at a specific position (1-based index)
void deleteAtPosition(Node** head, int position) {
    if (*head == NULL || position < 1) return;
    Node* temp = *head;

    // Delete head
    if (position == 1) {
        *head = temp->next;
        free(temp);
        return;
    }

    // Find previous node of the node to be deleted
    for (int i = 1; i < position - 1 && temp != NULL; i++)
        temp = temp->next;

    if (temp == NULL || temp->next == NULL)
        return;

    Node* toDelete = temp->next;
    temp->next = toDelete->next;
    free(toDelete);
}

// Print GPA list
void printList(Node* head) {
    while (head != NULL) {
        printf("GPA: %.1f\n", head->gpa);
        head = head->next;
    }
}

// Free memory
void freeList(Node* head) {
    Node* temp;
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }
}

int main() {
    int n, pos;
    float gpa;
    Node* head = NULL;

    scanf("%d", &n);

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

    scanf("%d", &pos);

    deleteAtPosition(&head, pos);
    printList(head);
    freeList(head);
    return 0;
}
```

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