Rajalakshmi Engineering College

Name: AJAY PRASATH
Email: 240701023@rajalakshmi.edu.in
Roll no: 240701023
Phone: 8778228414
Branch: REC
Department: I CSE AG
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 2

Attempt : 3
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Arun is learning about data structures and algorithms. He needs your help in solving a specific problem related to a singly linked list.

Your task is to implement a program to delete a node at a given position. If the position is valid, the program should perform the deletion; otherwise, it should display an appropriate message.

Input Format

The first line of input consists of an integer N, representing the number of elements in the linked list.

The second line consists of N space-separated elements of the linked list.

The third line consists of an integer x, representing the position to delete.

Position starts from 1.

Output Format

The output prints space-separated integers, representing the updated linked list after deleting the element at the given position.

If the position is not valid, print "Invalid position. Deletion not possible."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
8 2 3 1 7
2
Output: 8 3 1 7

Answer

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

void insert(int);
void display_List();
void deleteNode(int);

struct node {
    int data;
    struct node* next;
} *head = NULL, *tail = NULL;

// You are using GCC


void insert(int value) {
    struct node* newNode = (struct node*)malloc(sizeof(struct node));
    newNode->data = value;
    newNode->next = NULL;

    if (head == NULL) {
        head = tail = newNode;
    } else {
        tail->next = newNode;
        tail = newNode;
    }
}

void display_List() {
    struct node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
}

void deleteNode(int position) {
    if (position < 1) {
        printf("Invalid position. Deletion not possible.\n");
        return;
    }

    struct node* temp = head;

    // Deleting first node
    if (position == 1) {
        if (head == NULL) {
            printf("Invalid position. Deletion not possible.\n");
            return;
        }
        head = head->next;
        free(temp);
        display_List();
        return;
    }

    // Traverse to node before the one to be deleted
    for (int i = 1; temp != NULL && i < position - 1; i++) {
        temp = temp->next;
    }

    // If position is more than number of nodes
    if (temp == NULL || temp->next == NULL) {
        printf("Invalid position. Deletion not possible.\n");
        return;
    }

    struct node* nodeToDelete = temp->next;
    temp->next = nodeToDelete->next;

    // Update tail if needed
    if (nodeToDelete == tail) {
        tail = temp;
    }

    free(nodeToDelete);
    display_List();
}


int main() {
    int num_elements, element, pos_to_delete;

    scanf("%d", &num_elements);

    for (int i = 0; i < num_elements; i++) {
        scanf("%d", &element);
        insert(element);
    }

    scanf("%d", &pos_to_delete);

    deleteNode(pos_to_delete);

    return 0;
}
```

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