

# Rajalakshmi Engineering College

Name: Sanjay Kumar K

Email: 241801244@rajalakshmi.edu.in

Roll no: 241801244

Phone: 9710199820

Branch: REC

Department: I 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 5

Attempt : 1

Total Mark : 10

Marks Obtained : 10

#### Section 1 : Coding

##### 1. Problem Statement

Ashwin is tasked with developing a simple application to manage a list of items in a shop inventory using a doubly linked list. Each item in the inventory has a unique identification number. The application should allow users to perform the following operations:

Create a List of Items: Initialize the inventory with a given number of items. Each item will be assigned a unique number provided by the user and insert the elements at end of the list.

Delete an Item: Remove an item from the inventory at a specific position.

Display the Inventory: Show the list of items before and after deletion.

If the position provided for deletion is invalid (e.g., out of range), it should

display an error message.

### ***Input Format***

The first line contains an integer  $n$ , representing the number of items to be initially entered into the inventory.

The second line contains  $n$  integers, each representing the unique identification number of an item separated by spaces.

The third line contains an integer  $p$ , representing the position of the item to be deleted from the inventory.

### ***Output Format***

The first line of output prints "Data entered in the list:" followed by the data values of each node in the doubly linked list before deletion.

If  $p$  is an invalid position, the output prints "Invalid position. Try again."

If  $p$  is a valid position, the output prints "After deletion the new list:" followed by the data values of each node in the doubly linked list after deletion.

Refer to the sample output for the formatting specifications.

### ***Sample Test Case***

Input: 4

1 2 3 4

5

Output: Data entered in the list:

node 1 : 1

node 2 : 2

node 3 : 3

node 4 : 4

Invalid position. Try again.

### ***Answer***

```
// You are using GCC
```

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

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

```
struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};
```

```
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        fprintf(stderr, "Memory allocation failed\n");
        exit(EXIT_FAILURE);
    }
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}
```

```
void insertAtEnd(struct Node** head, struct Node** tail, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        *tail = newNode;
    } else {
        newNode->prev = *tail;
        (*tail)->next = newNode;
        *tail = newNode;
    }
}
```

```
int deleteAtPosition(struct Node** head, struct Node** tail, int position) {
    if (*head == NULL || position < 1) {
        return 0;
    }
    struct Node* current = *head;
    int count = 1;
    while (current && count < position) {
        current = current->next;
        count++;
    }
    if (!current) {
```

```

    return 0;
}
if (current == *head) {
    *head = current->next;
    if (*head) {
        (*head)->prev = NULL;
    } else {
        *tail = NULL;
    }
} else if (current == *tail) {
    *tail = current->prev;
    if (*tail) {
        (*tail)->next = NULL;
    }
} else {
    current->prev->next = current->next;
    current->next->prev = current->prev;
}
free(current);
return 1;
}

```

```

void displayList(struct Node* head, const char* message) {
    printf("%s\n", message);
    struct Node* current = head;
    int count = 1;
    while (current != NULL) {
        printf(" node %d : %d\n", count++, current->data);
        current = current->next;
    }
}

```

```

void freeList(struct Node* head) {
    struct Node* current = head;
    struct Node* nextNode;
    while (current != NULL) {
        nextNode = current->next;
        free(current);
        current = nextNode;
    }
}

```

```

int main() {
    int n, item, p;
    struct Node* head = NULL;
    struct Node* tail = NULL;

    if (scanf("%d", &n) != 1) return 1;
    for (int i = 0; i < n; i++) {
        if (scanf("%d", &item) != 1) {
            freeList(head);
            return 1;
        }
        insertAtEnd(&head, &tail, item);
    }
    displayList(head, "Data entered in the list:");

    if (scanf("%d", &p) != 1) {
        freeList(head);
        return 1;
    }

    if (p >= 1 && p <= n) {
        if (deleteAtPosition(&head, &tail, p)) {
            displayList(head, " After deletion the new list:");
        } else {
            printf("Invalid position. Try again.\n");
        }
    } else {
        printf("Invalid position. Try again.\n");
    }

    freeList(head);
    return 0;
}

```

**Status :** Correct

**Marks :** 10/10