

# Rajalakshmi Engineering College

Name: Rakesh H  
Email: 240701415@rajalakshmi.edu.in  
Roll no: 240701415  
Phone: 7305737702  
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 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>
```

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

```
typedef struct DoublyLinkedList {  
    Node* head;  
} DoublyLinkedList;
```

```
Node* createNode(int data) {  
    Node* newNode = (Node*)malloc(sizeof(Node));  
    newNode->data = data;  
    newNode->prev = NULL;  
    newNode->next = NULL;  
    return newNode;  
}
```

```
void append(DoublyLinkedList* list, int data) {  
    Node* newNode = createNode(data);  
    if (list->head == NULL) {  
        list->head = newNode;  
        return;  
    }  
    Node* temp = list->head;  
    while (temp->next != NULL) {  
        temp = temp->next;  
    }  
    temp->next = newNode;  
    newNode->prev = temp;  
}
```

```
void deleteNode(DoublyLinkedList* list, int position) {  
    if (list->head == NULL) {  
        printf("Invalid position. Try again.\n");  
        return;  
    }  
    Node* temp = list->head;  
    int count = 1;  
    while (temp != NULL && count < position) {
```

```

        temp = temp->next;
        count++;
    }

    if (temp == NULL) {
        printf("Invalid position. Try again.\n");
        return;
    }

    if (temp->prev != NULL) {
        temp->prev->next = temp->next;
    } else {
        list->head = temp->next;
    }

    if (temp->next != NULL) {
        temp->next->prev = temp->prev;
    }

    free(temp);
}

void display(DoublyLinkedList* list) {
    Node* temp = list->head;
    int i=1;
    while (temp != NULL) {
        printf("node %d : %d \n",i,temp->data);
        temp = temp->next;
        i++;
    }
    printf("\n");
}

int main() {
    DoublyLinkedList list;
    list.head = NULL;

    int n, p;

    scanf("%d", &n);
    int items[n];

```

```
for (int i = 0; i < n; i++) {  
    scanf("%d", &items[i]);  
    append(&list, items[i]);  
}  
  
// Display initial inventory  
printf("Data entered in the list:\n");  
display(&list);  
  
scanf("%d", &p);  
  
if (p < 1 || p > n) {  
    printf("Invalid position. Try again.\n");  
} else {  
    deleteNode(&list, p);  
    printf("After deletion the new list:\n");  
    display(&list);  
}  
  
return 0;  
}
```

**Status :** Correct

**Marks :** 10/10