Rajalakshmi Engineering College

Name: Sowmya R
Email: 241501207@rajalakshmi.edu.in
Roll no: 241501207
Phone: 9677182021
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 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

```
#include<stdio.h>
#include<stdlib.h>
typedef struct node{
```

```
    int data;
    struct node*next;
    struct node*back;
}node;
node*insert(node*head,int data){
    node*newnode=(node*)malloc(sizeof(node));
    newnode->data=data;
    newnode->next=NULL;
    newnode->back=NULL;

    if(!head)return newnode;
    node*temp=head;
    while(temp->next){
        temp=temp->next;
    }
    temp->next=newnode;
    newnode->back = temp;
    return head;
}
node*del(node*head,int pos){
    if(!head)return head;
    node*temp=head;

    if(pos==1){
        head=head->next;
        if(head)head->back=NULL;
        free(temp);
        return head;
    }
    for(int i=1;temp && i!=pos;i++){
        temp=temp->next;
    }
    if(!temp)return head;
    if(temp->next)temp->next->back = temp->back;
    if(temp->back)temp->back->next=temp->next;

    free(temp);
    return head;
}

void print(node*head){
    node*temp=head;
```

```
    int i=1;
    while(temp){
        printf("node %d : %d\n",i++,temp->data);
        temp=temp->next;
    }
}

void freelist(node*head){
    node*temp=head;
    while(head){
        temp=head;
        head=head->next;
        free(temp);
    }
}

int main(){
    node*head = NULL;
    int n;
    scanf("%d",&n);

    for(int i=0;i<n;i++){
        int d;
        scanf("%d",&d);
        head=insert(head,d);

    }
    printf("Data entered in the list:\n");
    print(head);

    int pos;
    scanf("%d",&pos);

    if(pos <=0 || pos > n){
        printf("Invalid position. Try again.");
        return 0;
    }
    head=del(head,pos);
    printf("\nAfter deletion the new list:\n");
    print(head);

    freelist(head);
```

```
    return 0;
}
```

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