Rajalakshmi Engineering College

Name: RANNESH KHUMAR B R
Email: 240701422@rajalakshmi.edu.in
Roll no: 2116240701
Phone: 9042350670
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_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Imagine Anu is tasked with finding the middle element of a doubly linked list. Given a doubly linked list where each node contains an integer value and is inserted at the end, implement a program to find the middle element of the list. If the number of nodes is even, return the middle element pair.

Input Format

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

The second line consists of N space-separated integers, representing the values of the nodes in the doubly linked list.

Output Format

The first line of output prints the space-separated elements of the doubly linked list.

The second line prints the middle element(s) of the doubly linked list, depending on whether the number of nodes is odd or even.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50

Output: 10 20 30 40 50
30

Answer

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

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

Node*head=NULL;
Node*tail=NULL;

void insert(int data){
   Node*newnode=(Node*)malloc(sizeof(Node));
   newnode->data=data;
   newnode->next=NULL;

   if(head==NULL){
      newnode->prev=NULL;
      head=newnode;
      tail=newnode;
   }
   else{
      tail->next=newnode;
      newnode->prev=tail;
```

```
        tail=newnode;
    }
}
void dis(){
    Node*temp=head;
    while(temp!=NULL){
        printf("%d ",temp->data);
        temp=temp->next;
    }
    printf("\n");
}

void pm(){
    Node*s1=head;
    Node*s2=head;

    while(s2!=NULL&&s2->next!=NULL){
        s1=s1->next;
        s2=s2->next->next;
    }
    if(s2==NULL){
        printf("%d %d\n",s1->prev->data,s1->data);
    }
    else{
        printf("%d\n",s1->data);
    }
}

int main(){
    int n,data;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d",&data);
        insert(data);
    }
    dis();
    pm();
}
```

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

2. Problem Statement

You are required to implement a program that deals with a doubly linked list.

The program should allow users to perform the following operations:

Insertion at the End: Insert a node with a given integer data at the end of the doubly linked list.Insertion at a given Position: Insert a node with a given integer data at a specified position within the doubly linked list.Display the List: Display the elements of the doubly linked list.

Input Format

The first line of input consists of an integer n, representing the number of elements to be initially inserted into the doubly linked list.

The second line consists of n space-separated integers, denoting the elements to be inserted at the end.

The third line consists of integer m, representing the new element to be inserted.

The fourth line consists of an integer p, representing the position at which the new element should be inserted (1-based indexing).

Output Format

If p is valid, display the elements of the doubly linked list after performing the insertion at the specified position.

If p is invalid, display "Invalid position" in the first line and the second line prints the original list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 25 34 48 57
35
4

Output: 10 25 34 35 48 57

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>

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

Node*create(int data){
    Node*newnode=(Node*)malloc(sizeof(Node));
    newnode->data=data;
    newnode->next=NULL;
    newnode->prev=NULL;
    return newnode;
}

void insert(Node**head,int data){
    Node*newnode=create(data);
    if(*head==NULL){
        *head=newnode;
        return;
    }
    Node*temp=*head;
    while(temp->next!=NULL){
        temp=temp->next;
    }
    temp->next=newnode;
    newnode->prev=temp;
}

void ins(Node**head,int data,int pos,int n){
    if(pos<1||pos>n+1){
        printf("Invalid position\n");
        return;
    }
    Node*newnode=create(data);
    if(pos==1){
```

```
        newnode->next=*head;
        if(*head!=NULL){
            (*head)->prev=newnode;}
            *head=newnode;
            return;
    }
    Node*temp=*head;
    for(int i=1;i<pos-1;i++){
        temp=temp->next;
    }
    newnode->next=temp->next;
    if(temp->next!=NULL)
    temp->next->prev=newnode;
    temp->next=newnode;
    newnode->prev=temp;
}

void dis(Node*head){
    Node*temp=head;
    while(temp!=NULL){
        printf("%d ",temp->data);
        temp=temp->next;
    }
    printf("\n");
}

int main(){
    int n,m,p;
    Node*head=NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        int data;
        scanf("%d",&data);
        insert(&head,data);
    }

    scanf("%d",&m);
    scanf("%d",&p);

    if(p<1||p>n+1){
        printf("Invalid position\n");
        dis(head);
```

```
    }
    else{
        ins(&head,m,p,n);
        dis(head);
    }
}
```

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

3. Problem Statement

Imagine you're managing a store's inventory list, and some products were accidentally entered multiple times. You need to remove the duplicate products from the list to ensure each product appears only once.

You have an unsorted doubly linked list of product IDs. Some of these product IDs may appear more than once, and your goal is to remove any duplicates.

Input Format

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

The second line of input consists of n space-separated integers representing the list elements.

Output Format

The output prints the final after removing duplicate nodes, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10
12 12 10 4 8 4 6 4 4 8
Output: 8 4 6 10 12

Answer

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

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

void insert(Node**head,int data){
    Node*newnode=(Node*)malloc(sizeof(Node));
    newnode->data=data;
    newnode->next=NULL;

    if(*head==NULL){
        newnode->prev=NULL;
        *head=newnode;
        return;
    }

    Node*temp=*head;
    while(temp->next){
        temp=temp->next;
    }
    temp->next=newnode;
    newnode->prev=temp;
}

void removedup(Node*head){
    int freq[101]={0};
    Node*temp=head;

    while(temp){
        freq[temp->data]++;
        temp=temp->next;
    }

    temp=head;
    while(temp){
        if(freq[temp->data]>1){
            freq[temp->data]--;
        }
```

```
        temp=temp->next;
    }

    temp=head;
    while(temp->next){
        temp=temp->next;
    }

    while(temp){
        if(freq[temp->data]==1){
            printf("%d ",temp->data);
            freq[temp->data]=0;
        }
        temp=temp->prev;
    }
    printf("\n");
}

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

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

    removedup(head);
}
```

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