Rajalakshmi Engineering College

Name: Naren S
Email: 240701346@rajalakshmi.edu.in
Roll no: 240701346
Phone: 7695937740
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

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node* prev,*next;
};

struct node* head=NULL;

void insert(int e){
    struct node* newnode=(struct node*)malloc(sizeof(node));
    newnode->data=e;
    newnode->next=NULL;
    newnode->prev=NULL;

    if(head==NULL){
        head=newnode;
    }
    else{
        struct node* temp=head;
        while(temp->next!=NULL){
            temp=temp->next;
        }
        temp->next=newnode;
        newnode->prev=temp;
```

```
    }
}

void findmid(int n){
    struct node* temp=head;
    if(n%2>0){
        int mid;
        mid=(n+1)/2;
        for(int i=1;i<mid;i++){
            temp=temp->next;
        }
        printf("%d",temp->data);
    }
    else{
        int mid;
        mid=int(n/2);
        for(int i=1;i<mid;i++){
            temp=temp->next;
        }
        printf("%d %d",temp->data,temp->next->data);
    }
}

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


int main(){
    int n;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        int e;
```

```
        scanf("%d",&e);
        insert(e);
    }
    traverse();
    printf("\n");
    findmid(n);
}
```

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

2. 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>

#define MAX 101  // Since product IDs range from 1 to 100

// Node structure for doubly linked list
typedef struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
} Node;

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;
    return newNode;
}

// Function to insert a node at the head of the list
void insertHead(Node** head, int data) {
    Node* newNode = createNode(data);
    newNode->next = *head;
    if (*head != NULL)
        (*head)->prev = newNode;
    *head = newNode;
}

// Function to remove duplicates from the list
void removeDuplicates(Node** head) {
    int seen[MAX] = {0};
    Node* current = *head;
    while (current != NULL) {
        if (seen[current->data]) {
            Node* toDelete = current;
            if (toDelete->prev)
                toDelete->prev->next = toDelete->next;
            if (toDelete->next)
                toDelete->next->prev = toDelete->prev;
```

```
            if (toDelete == *head)
                *head = toDelete->next;
            current = toDelete->next;
            free(toDelete);
        } else {
            seen[current->data] = 1;
            current = current->next;
        }
    }
}

// Function to print the list
void printList(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
        printf("%d", temp->data);
        if (temp->next != NULL)
            printf(" ");
        temp = temp->next;
    }
    printf("\n");
}

// Main function
int main() {
    int n;
    scanf("%d", &n);

    Node* head = NULL;

    // Insert at head to reverse order of first appearance
    for (int i = 0; i < n; i++) {
        int val;
        scanf("%d", &val);
        insertHead(&head, val);
    }

    removeDuplicates(&head);
    printList(head);

    return 0;
}
```

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

3. Problem Statement

Sam is learning about two-way linked lists. He came across a problem where he had to populate a two-way linked list and print the original as well as the reverse order of the list. Assist him with a suitable program.

Input Format

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

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

Output Format

The first line displays the message: "List in original order:"

The second line displays the elements of the doubly linked list in the original order.

The third line displays the message: "List in reverse order:"

The fourth line displays the elements of the doubly linked list in reverse order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5
Output: List in original order:
1 2 3 4 5
List in reverse order:
5 4 3 2 1

Answer

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

```c
#include <stdlib.h>

// Structure for a doubly linked list node
typedef struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
} Node;

// Function to create a new node
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}

// Function to insert a new node at the end of the list
void insertEnd(Node** head, int data) {
    Node* newNode = createNode(data);

    // If the list is empty, new node becomes the head
    if (*head == NULL) {
        *head = newNode;
    } else {
        Node* temp = *head;
        // Traverse to the last node
        while (temp->next != NULL) {
            temp = temp->next;
        }
        // Insert new node at the end
        temp->next = newNode;
        newNode->prev = temp;
    }
}

// Function to print the list in original order
void printOriginalOrder(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
        printf("%d", temp->data);
```

```
        if (temp->next != NULL) {
            printf(" ");
        }
        temp = temp->next;
    }
    printf("\n");
}

// Function to print the list in reverse order
void printReverseOrder(Node* head) {
    // Traverse to the last node first
    Node* temp = head;
    while (temp != NULL && temp->next != NULL) {
        temp = temp->next;
    }

    // Now print from last to first
    while (temp != NULL) {
        printf("%d", temp->data);
        if (temp->prev != NULL) {
            printf(" ");
        }
        temp = temp->prev;
    }
    printf("\n");
}

// Main function to execute the program
int main() {
    int n;
    scanf("%d", &n);

    Node* head = NULL;

    // Read the elements and populate the doubly linked list
    for (int i = 0; i < n; i++) {
        int data;
        scanf("%d", &data);
        insertEnd(&head, data);
    }

    // Print the list in original order
```

```
    printf("List in original order:\n");
    printOriginalOrder(head);

    // Print the list in reverse order
    printf("List in reverse order:\n");
    printReverseOrder(head);

    return 0;
}
```

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