

# Rajalakshmi Engineering College

Name: Sam Devaraja J  
Email: 240701463@rajalakshmi.edu.in  
Roll no: 2116240701463  
Phone: 7395954829  
Branch: REC  
Department: I CSE FE  
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>
typedef struct Node{
    int data;
    struct Node* prev;
    struct Node* next;
}Node;
Node* createNode(int data){
    Node* newNode=(Node*)malloc(sizeof(Node));
    newNode->data=data;
    newNode->prev=newNode->next=NULL;
    return newNode;
}
void printlist(Node* head){
    Node* temp=head;
    while(temp){
        printf("%d ",temp->data);
        temp=temp->next;
    }
    printf("\n");
}
void printmiddle(Node* head,int n){
    Node* temp=head;
    int mid=n/2;
    for(int i=0;i<mid;i++)
```

```

    {
        temp=temp->next;
    }
    if (n%2==0)
    {
        printf("%d %d\n",temp->prev->data,temp->data);
    }
    else
    {
        printf("%d\n",temp->data);
    }
}
int main()
{
    int n,val;
    scanf("%d",&n);
    Node* head=NULL,*tail=NULL;
    for(int i=0;i<n;i++){
        scanf("%d",&val);
        Node* newNode=createNode(val);
        if(!head){
            head=tail=newNode;
        }
        else
        {
            tail->next=newNode;
            newNode->prev=tail;
            tail=newNode;
        }
    }
    printlist(head);
    printmiddle(head,n);
    return 0;
}

```

**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***

```
// You are using GCC
#include <stdio.h>
int main() {
    int n;
    scanf("%d", &n);

    int arr[n], i, j, k = 0;
    int result[n];
    int seen[101] = {0};
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    for (i = n - 1; i >= 0; i--) {
```

```
        if (!seen[arr[i]]) {  
            result[k++] = arr[i];  
            seen[arr[i]] = 1;  
        }  
    }  
  
    for (i = 0; i < k; i++) {  
        printf("%d ", result[i]);  
    }  
  
    return 0;  
}
```

**Status :** Correct

**Marks :** 10/10

### 3. Problem Statement

Vanessa is learning about the doubly linked list data structure and is eager to play around with it. She decides to find out how the elements are inserted at the beginning and end of the list.

Help her implement a program for the same.

#### **Input Format**

The first line of input contains an integer N, representing the size of the doubly linked list.

The next line contains N space-separated integers, each representing the values to be inserted into the doubly linked list.

#### **Output Format**

The first line of output prints the integers, after inserting them at the beginning, separated by space.

The second line prints the integers, after inserting at the end, separated by space.

Refer to the sample output for formatting specifications.

### **Sample Test Case**

Input: 5

1 2 3 4 5

Output: 5 4 3 2 1

1 2 3 4 5

### **Answer**

// You are using GCC

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

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

```
struct Node {
```

```
    int data;
```

```
    struct Node* prev;
```

```
    struct Node* next;
```

```
};
```

```
void insertAtBeginning(struct Node** head, int data) {
```

```
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
```

```
    newNode->data = data;
```

```
    newNode->prev = NULL;
```

```
    newNode->next = *head;
```

```
    if (*head != NULL)
```

```
        (*head)->prev = newNode;
```

```
    *head = newNode;
```

```
}
```

```
void insertAtEnd(struct Node** head, int data) {
```

```
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
```

```
    newNode->data = data;
```

```
    newNode->next = NULL;
```

```
    if (*head == NULL) {
```

```
        newNode->prev = NULL;
```

```
        *head = newNode;
```

```
        return;
```

```
    }
```

```
    struct Node* temp = *head;
```

```
    while (temp->next != NULL)
```

```
        temp = temp->next;
```

```
    temp->next = newNode;
```

```
    newNode->prev = temp;
```

```
}
```

```
void display(struct Node* head) {
```

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

int main() {
    int N, i;
    scanf("%d", &N);
    int arr[10];
    for (i = 0; i < N; i++) {
        scanf("%d", &arr[i]);
    }
    struct Node* head1 = NULL;
    for (i = 0; i < N; i++) {
        insertAtBeginning(&head1, arr[i]);
    }
    display(head1);
    struct Node* head2 = NULL;
    for (i = 0; i < N; i++) {
        insertAtEnd(&head2, arr[i]);
    }
    display(head2);
    return 0;
}
```

**Status :** Correct

**Marks : 10/10**