

Rajalakshmi Engineering College

Name: akshaya d
Email: 241901002@rajalakshmi.edu.in
Roll no: 241901002
Phone: 9363744696
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 1

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Your task is to create a program to manage a playlist of items. Each item is represented as a character, and you need to implement the following operations on the playlist.

Here are the main functionalities of the program:

Insert Item: The program should allow users to add items to the front and end of the playlist. Items are represented as characters. Display Playlist: The program should display the playlist containing the items that were added.

To implement this program, a doubly linked list data structure should be used, where each node contains an item character.

Input Format

The input consists of a sequence of space-separated characters, representing the items to be inserted into the doubly linked list.

The input is terminated by entering - (hyphen).

Output Format

The first line of output prints "Forward Playlist: " followed by the linked list after inserting the items at the end.

The second line prints "Backward Playlist: " followed by the linked list after inserting the items at the front.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: a b c -

Output: Forward Playlist: a b c

Backward Playlist: c b a

Answer

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

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

```
struct Node {  
    char item;  
    struct Node* next;  
    struct Node* prev;  
};
```

```
// You are using GCC
```

```
void insertAtEnd(struct Node** head, char item) {  
    struct Node* newNode=(struct Node*)malloc(sizeof(struct Node));  
    newNode->item=item;  
    newNode->next=NULL;  
    if(*head==NULL){  
        newNode->prev=NULL;  
        *head=newNode;  
        return;  
    }  
}
```

```

    struct Node* last=*head;
    while(last->next!=NULL){
        last=last->next;
    }
    last->next=newNode;
    newNode->prev=last;
}
void displayForward(struct Node* head) {
    struct Node*current=head;
    while(current!=NULL){
        printf("%c ",current->item);
        current=current->next;
    }
    printf("\n");
}

```

```

void displayBackward(struct Node* tail) {
    struct Node*current=tail;
    if(current==NULL)
        return;
    while(current->next!=NULL){
        current=current->next;
    }
    while(current!=NULL){
        printf("%c ",current->item);
        current=current->prev;
    }
}

```

```

void freePlaylist(struct Node* head) {
    Node*current=head;
    Node*nextNode;
    while(current!=NULL){
        nextNode=current->next;
        free(current);
        current=nextNode;
    }
}

```

```
int main() {
    struct Node* playlist = NULL;
    char item;

    while (1) {
        scanf(" %c", &item);
        if (item == '-') {
            break;
        }
        insertAtEnd(&playlist, item);
    }

    struct Node* tail = playlist;
    while (tail->next != NULL) {
        tail = tail->next;
    }

    printf("Forward Playlist: ");
    displayForward(playlist);

    printf("Backward Playlist: ");
    displayBackward(tail);

    freePlaylist(playlist);

    return 0;
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: akshaya d
Email: 241901002@rajalakshmi.edu.in
Roll no: 241901002
Phone: 9363744696
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Moniksha, a chess coach organizing a tournament, needs a program to manage participant IDs efficiently. The program maintains a doubly linked list of IDs and offers two functions: Append to add IDs as students register, and Print Maximum ID to identify the highest ID for administrative tasks.

This tool streamlines tournament organization, allowing Moniksha to focus on coaching her students effectively.

Input Format

The first line consists of an integer n , representing the number of participant IDs to be added.

The second line consists of n space-separated integers representing the participant IDs.

Output Format

The output displays a single integer, representing the maximum participant ID.

If the list is empty, the output prints "Empty list!".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3

163 137 155

Output: 163

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct Node{
    int id;
    struct Node*next;
    struct Node*prev;
};
struct DoublyLinkedList{
    struct Node*head;
    struct Node*tail;
};
struct DoublyLinkedList*createList(){
    struct DoublyLinkedList*list=(struct DoublyLinkedList*)malloc(sizeof(struct
DoublyLinkedList));
    list->head=NULL;
    list->tail=NULL;
    return list;
}
void append(struct DoublyLinkedList*list,int id){
    struct Node*newNode=(struct Node*)malloc(sizeof(struct Node));
    newNode->id=id;
    newNode->next=NULL;
    if(list->head==NULL){
```

```

        newNode->prev=NULL;
        list->head=newNode;
        list->tail=newNode;
        return;
    }
    newNode->prev=list->tail;
    list->tail->next=newNode;
    list->tail=newNode;

}

int printMaxID(struct DoublyLinkedList*list){
    if(list->head==NULL){
        return -1;
    }
    int maxID=list->head->id;
    struct Node*current=list->head;
    while(current!=NULL){
        if(current->id>maxID){
            maxID=current->id;
        }
        current=current->next;
    }return maxID;
}

int main(){
    int n;
    scanf("%d",&n);
    struct DoublyLinkedList*list=createList();
    for(int i=0;i<n;i++){
        int id;
        scanf("%d",&id);
        append(list,id);
    }
    int maxID=printMaxID(list);
    if(maxID==-1){
        printf("Empty list!\n");

    }
    else{
        printf("%d\n",maxID);
    }
    return 0;
}

```

}

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: akshaya d
Email: 241901002@rajalakshmi.edu.in
Roll no: 241901002
Phone: 9363744696
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Bob is tasked with developing a company's employee record management system. The system needs to maintain a list of employee records using a doubly linked list. Each employee is represented by a unique integer ID.

Help Bob to complete a program that adds employee records at the front, traverses the list, and prints the same for each addition of employees to the list.

Input Format

The first line of input consists of an integer N, representing the number of employees.

The second line consists of N space-separated integers, representing the employee IDs.

Output Format

For each employee ID, the program prints "Node Inserted" followed by the current state of the doubly linked list in the next line, with the data values of each node separated by spaces.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4

101 102 103 104

Output: Node Inserted

101

Node Inserted

102 101

Node Inserted

103 102 101

Node Inserted

104 103 102 101

Answer

```
#include <iostream>
using namespace std;
```

```
struct node {
    int info;
    struct node* prev, * next;
};
```

```
struct node* start = NULL;
```

```
// You are using GCC
```

```
void traverse() {
    struct node* current=start;
    while(current!=NULL){
        printf("%d\n",current->info);
        current=current->next;
    }
}
```

```
void insertAtFront(int data) {  
    struct node* newNode=(struct node*)malloc(sizeof(struct node));  
    newNode->info=data;  
    newNode->next=start;  
    newNode->prev=NULL;  
    if(start!=NULL)start->prev=newNode;  
    start=newNode;  
    printf("Node inserted\n");  
}  
  
int main() {  
    int n, data;  
    cin >> n;  
    for (int i = 0; i < n; ++i) {  
        cin >> data;  
        insertAtFront(data);  
        traverse();  
    }  
    return 0;  
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: akshaya d
Email: 241901002@rajalakshmi.edu.in
Roll no: 241901002
Phone: 9363744696
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Ravi is developing a student registration system for a college. To efficiently store and manage the student IDs, he decides to implement a doubly linked list where each node represents a student's ID.

In this system, each student's ID is stored sequentially, and the system needs to display all registered student IDs in the order they were entered.

Implement a program that creates a doubly linked list, inserts student IDs, and displays them in the same order.

Input Format

The first line contains an integer N the number of student IDs.

The second line contains N space-separated integers representing the student IDs.

Output Format

The output should display the single line containing N space-separated integers representing the student IDs stored in the doubly linked list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

10 20 30 40 50

Output: 10 20 30 40 50

Answer

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

struct doublyLinkedList{
    struct node* head;
    struct node* tail;
};

struct doublyLinkedList* createList(){
    struct doublyLinkedList* list=(struct doublyLinkedList*)malloc(sizeof (struct
doublyLinkedList));
    list->head=NULL;
    list->tail=NULL;
    return list;
}

void insert(struct doublyLinkedList* list,int id){
    struct node*newnode=(struct node*)malloc(sizeof(struct node));
    newnode->id=id;
    newnode->next=NULL;
```

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

void display(struct doublyLinkedList*list){
    struct node*current=list->head;
    while(current!=NULL){
        printf("%d ",current->id);
        current=current->next;
    }
}

int main(){
    int N;
    scanf("%d",&N);
    struct doublyLinkedList*list=createList();
    for(int i=0;i<N;i++){
        int id;
        scanf("%d",&id);
        insert(list,id);
    }
    display(list);
    printf("\n");
    return 0;
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: akshaya d
Email: 241901002@rajalakshmi.edu.in
Roll no: 241901002
Phone: 9363744696
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

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>
```



```

struct node{
    int id;
    struct node*next;
    struct node* prev;

};
struct DoublyLinkedList{
    struct node*head;
    struct node* tail;

};
struct DoublyLinkedList* createList(){
    struct DoublyLinkedList*list=(struct DoublyLinkedList*)malloc(sizeof(struct
DoublyLinkedList));
    list->head=NULL;
    list->tail=NULL;
    return list;

}
void insert(struct DoublyLinkedList*list,int id){
    struct node*newnode=(struct node*)malloc(sizeof(struct node));
    newnode->id=id;
    newnode->next=NULL;
    if(list->head==NULL){
        newnode->prev=NULL;
        list->head=newnode;
        list->tail=newnode;
        return;
    }
    newnode->prev=list->tail;
    list->tail->next=newnode;
    list->tail=newnode;

}
void display(struct DoublyLinkedList*list){
    struct node*current=list->head;
    int index=1;
    while(current!=NULL){
        printf("node %d : %d\n",index,current->id);
        current=current->next;
        index++;
    }
}

```

```

}
void deletenode(struct DoublyLinkedList*list,int position){
    if(position<1){
        printf("Invalid position.Try again.\n");
        return;
    }
    struct node*current=list->head;
    for(int i=1;i<position&&current!=NULL;i++){
        current=current->next;
    }
    if(current==NULL){
        printf("Invalid position. Try again.\n");
        return;
    }
    if(current->prev!=NULL){
        current->prev->next=current->next;
    }
    else{
        list->head=current->next;
    }
    if(current->next!=NULL){
        current->next->prev=current->prev;
    }
    else{
        list->tail=current->prev;
    }
    free(current);
}
int main(){
    int n,p;
    scanf("%d",&n);
    struct DoublyLinkedList*list=createList();
    for(int i=0;i<n;i++){
        int id;
        scanf("%d",&id);
        insert(list,id);
    }
    printf("Data entered in the list:\n");
    display(list);
    scanf("%d",&p);
    deletenode(list,p);
}

```

```
if(p>=1&&p<=n){  
    printf("After deletion the new list:\n");  
    display(list);  
}  
return 0;  
}
```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: akshaya d
Email: 241901002@rajalakshmi.edu.in
Roll no: 241901002
Phone: 9363744696
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

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

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

```
// You are using GCC
```

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

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

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

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

```
struct DoublyLinkedList* createList() {  
    struct DoublyLinkedList* list = (struct DoublyLinkedList*)malloc(sizeof(struct  
DoublyLinkedList));  
    list->head = NULL;  
    return list;  
}
```

```
void insertEnd(struct DoublyLinkedList* list, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    if (list->head == NULL) {
        newNode->prev = NULL;
        list->head = newNode;
        return;
    }
    struct Node* last = list->head;
    while (last->next != NULL) {
        last = last->next;
    }
    last->next = newNode;
    newNode->prev = last;
}
```

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

```
void displayReverse(struct DoublyLinkedList* list) {
    struct Node* temp = list->head;
    if (temp == NULL) return;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->prev;
    }
    printf("\n");
}
```

```
int main() {
    int n;
    scanf("%d", &n);
```

```
struct DoublyLinkedList* list = createList();
for (int i = 0; i < n; i++) {
    int value;
    scanf("%d", &value);
    insertEnd(list, value);
}
printf("List in original order:\n");
displayOriginal(list);
printf("List in reverse order:\n");
displayReverse(list);
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 list 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>
```

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

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

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

```
void insertAtBeginning(struct Node** head, int data) {  
    struct Node* newNode = createNode(data);  
    if (*head == NULL) {  
        *head = newNode;  
        return;  
    }  
    newNode->next = *head;  
    (*head)->prev = newNode;  
    *head = newNode;  
}
```

```
void insertAtEnd(struct Node** head, int data) {  
    struct Node* newNode = createNode(data);  
    if (*head == NULL) {  
        *head = newNode;  
        return;  
    }  
}
```



```

    struct Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->prev = temp;
}

void removeDuplicates(struct Node** head) {
    struct Node* current = *head;
    struct Node* newHead = NULL;
    int lastOccurrence[101] = {-1};
    int index = 0;
    current = *head;
    while (current != NULL) {
        lastOccurrence[current->data] = index;
        current = current->next;
        index++;
    }
    current = *head;
    index = 0;
    while (current != NULL) {
        if (lastOccurrence[current->data] == index) {
            insertAtBeginning(&newHead, current->data);
        }
        current = current->next;
        index++;
    }
    struct Node* reversedHead = NULL;
    current = newHead;
    while (current != NULL) {
        insertAtEnd(&reversedHead, current->data);
        current = current->next;
    }
    *head = reversedHead;
}

```

```

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

```

```

    }
    printf("\n");
}

int main() {
    int n;
    scanf("%d", &n);
    struct Node* head = NULL;
    int value;
    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        insertAtEnd(&head, value);
    }
    removeDuplicates(&head);
    printList(head);
    return 0;
}

```

Status : Correct

Marks : 10/10

3. 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>
```

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

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

```
struct DoublyLinkedList* createList() {  
    struct DoublyLinkedList* list = (struct DoublyLinkedList*)malloc(sizeof(struct  
DoublyLinkedList));  
    list->head = NULL;  
    return list;
```

```
}
```

```
void insertEnd(struct DoublyLinkedList* list, int data) {  
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));  
    newNode->data = data;  
    newNode->next = NULL;  
    if (list->head == NULL) {  
        newNode->prev = NULL;  
        list->head = newNode;  
        return;  
    }  
    struct Node* last = list->head;  
    while (last->next != NULL) {  
        last = last->next;  
    }  
    last->next = newNode;  
    newNode->prev = last;  
}
```

```
void insertAtPosition(struct DoublyLinkedList* list, int data, int position) {  
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));  
    newNode->data = data;  
    if (position == 1) {  
        newNode->next = list->head;  
        newNode->prev = NULL;  
        if (list->head != NULL) {  
            list->head->prev = newNode;  
        }  
        list->head = newNode;  
        return;  
    }  
    struct Node* temp = list->head;  
    for (int i = 1; temp != NULL && i < position - 1; i++) {  
        temp = temp->next;  
    }  
    if (temp == NULL) {  
        free(newNode);  
        return;  
    }  
    newNode->next = temp->next;  
    temp->next = newNode;  
    newNode->prev = temp;
```

```

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

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

int main() {
    int n;
    scanf("%d", &n);
    struct DoublyLinkedList* list = createList();
    for (int i = 0; i < n; i++) {
        int value;
        scanf("%d", &value);
        insertEnd(list, value);
    }
    int m, p;
    scanf("%d", &m);
    scanf("%d", &p);
    if (p < 1 || p > n + 1) {
        printf("Invalid position\n");
        displayList(list);
    } else {
        insertAtPosition(list, m, p);
        displayList(list);
    }
    return 0;
}

```

Status : Correct

Marks : 10/10

Rajalakshmi Engineering College

Name: akshaya d
Email: 241901002@rajalakshmi.edu.in
Roll no: 241901002
Phone: 9363744696
Branch: REC
Department: I CSE (CS) FA
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_PAH

Attempt : 1
Total Mark : 50
Marks Obtained : 50

Section 1 : Coding

1. Problem Statement

Rohan is a software developer who is working on an application that processes data stored in a Doubly Linked List. He needs to implement a feature that finds and prints the middle element(s) of the list. If the list contains an odd number of elements, the middle element should be printed. If the list contains an even number of elements, the two middle elements should be printed.

Help Rohan by writing a program that reads a list of numbers, prints the list, and then prints the middle element(s) based on the number of elements in the list.

Input Format

The first line of the input consists of an integer n the number of elements in the

doubly linked list.

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

Output Format

The first line prints the elements of the list separated by space. (There is an extra space at the end of this line.)

The second line prints the middle element(s) based on the number of elements.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

20 52 40 16 18

Output: 20 52 40 16 18

40

Answer

```
// You are using GCC
```

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

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

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

```
struct Node* head = NULL;
```

```
// Function to insert a new node at the end of the list
```

```
void insert(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* last = head;
while (last->next != NULL) {
    last = last->next;
}

last->next = newNode;
newNode->prev = last;
}

// Function to print the list
void printList() {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n"); // Print a newline at the end of the list
}

// Function to find and print the middle element(s)
void printMiddle() {
    struct Node* slow = head;
    struct Node* fast = head;

    // Move fast pointer two nodes and slow pointer one node
    while (fast != NULL && fast->next != NULL) {
        fast = fast->next->next;
        slow = slow->next;
    }

    // If fast is NULL, the number of nodes is even
    if (fast == NULL) {
        // Print the two middle nodes
        printf("%d %d\n", slow->prev->data, slow->data);
    } else { // If fast is not NULL, the number of nodes is odd
        // Print the middle node
        printf("%d\n", slow->data);
    }
}

```



```

    }
}

int main() {
    int n;

    // Read the number of elements
    scanf("%d", &n);
    int element;

    // Read elements and insert them into the doubly linked list
    for (int i = 0; i < n; i++) {
        scanf("%d", &element);
        insert(element);
    }

    // Print the list
    printList();

    // Print the middle element(s)
    printMiddle();

    // Free the allocated memory
    struct Node* current = head;
    struct Node* nextNode;
    while (current != NULL) {
        nextNode = current->next;
        free(current);
        current = nextNode;
    }

    return 0;
}

```

Status : Correct

Marks : 10/10

2. Problem Statement

Tom is a software developer working on a project where he has to check if a doubly linked list is a palindrome. He needs to write a program to solve

this problem. Write a program to help Tom check if a given doubly linked list is a palindrome or not.

Input Format

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

The second line consists of N space-separated integers representing the linked list elements.

Output Format

The first line displays the space-separated integers, representing the doubly linked list.

The second line displays one of the following:

1. If the doubly linked list is a palindrome, print "The doubly linked list is a palindrome".
2. If the doubly linked list is not a palindrome, print "The doubly linked list is not a palindrome".

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5

1 2 3 2 1

Output: 1 2 3 2 1

The doubly linked list is a palindrome

Answer

```
// You are using GCC
```

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

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

```
// Define the structure for a Node in the doubly linked list
```

```
struct Node {
```

```
    int data;           // Data stored in the node
```

```
    struct Node* next;  // Pointer to the next node
```

```

    struct Node* prev;    // Pointer to the previous node
};

// Global head pointer for the doubly linked list
struct Node* head = NULL;

// Function to insert a new node at the end of the list
void insert(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* last = head;
    while (last->next != NULL) {
        last = last->next;
    }

    last->next = newNode;
    newNode->prev = last;
}

// Function to print the list
void printList() {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n"); // Print a newline at the end of the list
}

// Function to check if the list is a palindrome
int isPalindrome() {
    struct Node* left = head;
    struct Node* right = head;

```

```

// Move the right pointer to the end of the list
while (right->next != NULL) {
    right = right->next;
}

// Compare elements from the start and end
while (left != NULL && right != NULL && left != right && left->prev != right) {
    if (left->data != right->data) {
        return 0; // Not a palindrome
    }
    left = left->next;
    right = right->prev;
}
return 1; // Is a palindrome
}

int main() {
    int n;

    // Read the number of elements
    scanf("%d", &n);
    int element;

    // Read elements and insert them into the doubly linked list
    for (int i = 0; i < n; i++) {
        scanf("%d", &element);
        insert(element);
    }

    // Print the list
    printList();

    // Check if the list is a palindrome
    if (isPalindrome()) {
        printf("The doubly linked list is a palindrome\n");
    } else {
        printf("The doubly linked list is not a palindrome\n");
    }

    // Free the allocated memory
    struct Node* current = head;
    struct Node* nextNode;

```

```
while (current != NULL) {  
    nextNode = current->next;  
    free(current);  
    current = nextNode;  
}  
  
return 0;  
}
```

Status : Correct

Marks : 10/10

3. Problem Statement

Bala is a student learning about the doubly linked list and its functionalities. He came across a problem where he wanted to create a doubly linked list by appending elements to the front of the list.

After populating the list, he wanted to delete the node at the given position from the beginning. Write a suitable code to help Bala.

Input Format

The first line contains an integer N, the number of elements in the doubly linked list.

The second line contains N integers separated by a space, the data values of the nodes in the doubly linked list.

The third line contains an integer X, the position of the node to be deleted from the doubly linked list.

Output Format

The first line of output displays the original elements of the doubly linked list, separated by a space.

The second line prints the updated list after deleting the node at the given position X from the beginning.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5

10 20 30 40 50

2

Output: 50 40 30 20 10

50 30 20 10

Answer

// You are using GCC

#include <stdio.h>

#include <stdlib.h>

```
struct Node {  
    int data;           // Data stored in the node  
    struct Node* next;  // Pointer to the next node  
    struct Node* prev;  // Pointer to the previous node  
};
```

```
struct Node* head = NULL;
```

```
void insertAtFront(int data) {  
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));  
    newNode->data = data;  
    newNode->next = head;  
    newNode->prev = NULL;  
  
    if (head != NULL) {  
        head->prev = newNode;  
    }  
    head = newNode;  
}
```

```
void deleteAtPosition(int position) {  
    if (head == NULL || position <= 0) {  
        return; // No deletion needed  
    }
```

```
    struct Node* current = head;  
    if (position == 1) {  
        head = current->next;
```

```

    if (head != NULL) {
        head->prev = NULL;
    }
    free(current); // Free the old head
    return;
}

for (int i = 1; current != NULL && i < position; i++) {
    current = current->next;
}

// If the position is out of bounds
if (current == NULL) {
    return;
}

// Update pointers to remove the current node
if (current->next != NULL) {
    current->next->prev = current->prev; // Update previous pointer of next node
}
if (current->prev != NULL) {
    current->prev->next = current->next; // Update next pointer of previous node
}

free(current); // Free the node
}

// Function to print the list
void printList() {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n"); // Print a newline at the end of the list
}

```

```

int main() {
    int N, X;

```

```

// Read the number of elements in the doubly linked list
scanf("%d", &N);
int element;

// Read elements and insert them at the front of the doubly linked list
for (int i = 0; i < N; i++) {
    scanf("%d", &element);
    insertAtFront(element);
}

// Print the original list
printList();

// Read the position of the node to be deleted
scanf("%d", &X);

// Delete the node at the specified position
deleteAtPosition(X);

// Print the updated list
printList();

// Free the allocated memory
struct Node* current = head;
struct Node* nextNode;
while (current != NULL) {
    nextNode = current->next;
    free(current);
    current = nextNode;
}

return 0;
}

```

Status : Correct

Marks : 10/10

4. Problem Statement

Riya is developing a contact management system where recently added contacts should appear first. She decides to use a doubly linked list to store contact IDs in the order they are added. Initially, new contacts are

inserted at the front of the list. However, sometimes she needs to insert a new contact at a specific position in the list based on priority.

Help Riya implement this system by performing the following operations:

Insert contact IDs at the front of the list as they are added. Insert a new contact at a given position in the list.

Input Format

The first line of input consists of an integer N, representing the initial size of the linked list.

The second line consists of N space-separated integers, representing the values of the linked list to be inserted at the front.

The third line consists of an integer position, representing the position at which the new value should be inserted (position starts from 1).

The fourth line consists of integer data, representing the new value to be inserted.

Output Format

The first line of output prints the original list after inserting initial elements to the front.

The second line prints the updated linked list after inserting the element at the specified position.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4

10 20 30 40

3

25

Output: 40 30 20 10

40 30 25 20 10

Answer

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

```
struct Node {
    int data;          // Data stored in the node
    struct Node* next; // Pointer to the next node
    struct Node* prev; // Pointer to the previous node
};
```

```
// Global head pointer for the doubly linked list
struct Node* head = NULL;
```

```
// Function to insert a new node at the front of the list
void insertAtFront(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = head;
    newNode->prev = NULL;

    if (head != NULL) {
        head->prev = newNode;
    }
    head = newNode;
}
```

```
// Function to insert a new node at a specific position
void insertAtPosition(int data, int position) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;

    if (position == 1) {
        insertAtFront(data);
        return;
    }

    struct Node* current = head;
    for (int i = 1; i < position - 1 && current != NULL; i++) {
        current = current->next;
    }

    if (current == NULL) {
```

```

        printf("Position out of bounds\n");
        free(newNode);
        return;
    }

    newNode->next = current->next;
    newNode->prev = current;

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

// Function to print the list
void printList() {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n"); // Print a newline at the end of the list
}

```

```

int main() {
    int N, position, data;

    // Read the initial size of the linked list
    scanf("%d", &N);
    int element;

    // Read elements and insert them at the front of the doubly linked list
    for (int i = 0; i < N; i++) {
        scanf("%d", &element);
        insertAtFront(element);
    }

    // Print the original list
    printList();
}

```

```

// Read the position to insert the new value
scanf("%d", &position);
// Read the new value to be inserted
scanf("%d", &data);

// Insert the new value at the specified position
insertAtPosition(data, position);

// Print the updated list
printList();

// Free the allocated memory
struct Node* current = head;
struct Node* nextNode;
while (current != NULL) {
    nextNode = current->next;
    free(current);
    current = nextNode;
}

return 0;
}

```

Status : Correct

Marks : 10/10

5. Problem Statement

Pranav wants to clockwise rotate a doubly linked list by a specified number of positions. He needs your help to implement a program to achieve this. Given a doubly linked list and an integer representing the number of positions to rotate, write a program to rotate the list clockwise.

Input Format

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

The second line consists of n space-separated linked list elements.

The third line consists of an integer k, representing the number of places to rotate the list.

Output Format

The output displays the elements of the doubly linked list after rotating it by k positions.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5

1 2 3 4 5

1

Output: 5 1 2 3 4

Answer

```
// You are using GCC
```

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

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

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

```
struct Node* head = NULL;
```

```
// Function to insert a new node at the end of the list
```

```
void insert(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* last = head;
```

```

    while (last->next != NULL) {
        last = last->next;
    }

    last->next = newNode;
    newNode->prev = last;
}

// Function to print the list
void printList() {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n"); // Print a newline at the end of the list
}

// Function to rotate the doubly linked list clockwise by k positions
void rotateClockwise(int k) {
    if (head == NULL || head->next == NULL || k == 0) {
        return; // No rotation needed
    }

    struct Node* current = head;
    struct Node* last = head;
    int count = 1;

    // Traverse to the end of the list and count the nodes
    while (last->next != NULL) {
        last = last->next;
        count++;
    }

    // If k is greater than count, reduce k
    k = k % count;

    // If k is 0 after modulo, no rotation is needed
    if (k == 0) {
        return;
    }
}

```

```
// Traverse to the (count - k)th node
current = head;
for (int i = 1; i < count - k; i++) {
    current = current->next;
}

// Update the head and pointers
struct Node* newHead = current->next;
current->next = NULL;
newHead->prev = NULL;
last->next = head;
head->prev = last;
head = newHead;
}
```

```
int main() {
    int n, k;
```

```
// Read the number of elements
scanf("%d", &n);
int element;
```

```
// Read elements and insert them into the doubly linked list
for (int i = 0; i < n; i++) {
    scanf("%d", &element);
    insert(element);
}
```

```
// Read the number of positions to rotate
scanf("%d", &k);
```

```
// Rotate the list
rotateClockwise(k);
```

```
// Print the rotated list
printList();
```

```
// Free the allocated memory
struct Node* current = head;
struct Node* nextNode;
```

```
while (current != NULL) {  
    nextNode = current->next;  
    free(current);  
    current = nextNode;  
}
```

```
return 0;  
}
```

Status : Correct

Marks : 10/10