Rajalakshmi Engineering College

Name: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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;
};
```

```
void insertAtEnd(struct Node** head, char item) {
    Node * temp=(Node *)malloc(sizeof(Node));
    temp->prev=NULL;
```

```
    temp->next=NULL;
    temp->item=item;
    if(*head==NULL)
    {
        *head=temp;
    }
    else
    {
        Node * p=*head;
        while(p->next!=NULL)
        {
            p=p->next;
        }
        p->next=temp;
        temp->prev=p;
    }
   //type your code here
}
void displayForward(struct Node* head) {
    //type your code here
    Node * p=head;
    while(p!=NULL)
    {
        printf("%c ",p->item);
        p=p->next;
    }
    printf("\n");
}

void displayBackward(struct Node* tail) {
    //type your code here
    Node * p=tail;
    while(p!=NULL)
    {
        printf("%c ",p->item);
        p=p->prev;
    }
    printf("\n");
}

void freePlaylist(struct Node* head) {
    //type your code here
```

```
    Node * p=head;
    while(p!=NULL)
    {
        Node *a=p->next;
        free(p);
    p=a;
}
}

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: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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

```
#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));
   if(newNode == NULL)
   {
      perror("Memory allocation failed");
      exit(EXIT_FAILURE);
   }
   newNode->data = data;
   newNode->next = NULL;
   newNode->prev = NULL;
   return newNode;
}
void append(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 printMaximumID(struct Node* head)
{
    if(head == NULL)
    {
        printf("Empty list!\n");
        return;
    }
    int maxID = head->data;
    struct Node* current = head->next;
    while(current != NULL)
    {
        if(current->data > maxID)
        {
            maxID = current->data;
        }
        current = current->next;
    }
    printf("%d\n",maxID);
}
void freeList(struct Node* head)
{
    struct Node* current = head;
    struct Node* nextNode;
    while(current != NULL)
    {
        nextNode = current->next;
        free(current);
        current = nextNode;
    }
}
```

```
int main()
{
    int n,id;
    scanf("%d",&n);
    struct Node* head = NULL;
    if(n>0)
    {
        for(int i=0; i<n; i++)
        {
            scanf("%d",&id);
            append(&head,id);
        }
    }
    printMaximumID(head);
    freeList(head);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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;

void traverse() {
   //type your code here
   printf("Node Inserted\n");
   struct node * a=start;
   while(a!=NULL)
   {
      printf("%d ",a->info);
      a=a->next;
   }
```

```
    printf("\n");
}

void insertAtFront(int data) {
    //type your code here
    struct node * temp=(node *)malloc(sizeof(node));
    temp->info=data;
    temp->next=NULL;
    temp->prev=NULL;

    if(start==NULL)
    {
        start=temp;
    }
    else
    {
        temp->next=start;
        temp->next->prev=temp;
    }
    start=temp;
}


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: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
   int studentID;
   struct Node* next;
   struct Node* prev;
};
struct Node* createNode(int studentID)
{
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   if(newNode == NULL)
   {
      printf("Memory allocation failed\n");
      exit(1);
   }
   newNode->studentID = studentID;
   newNode->next = NULL;
   newNode->prev = NULL;
   return newNode;
}
void insertAtEnd(struct Node** head,struct Node** tail,int studentID)
{
   struct Node* newNode = createNode(studentID);
```

```
    if(*head == NULL)
    {
        *head = newNode;
        *tail = newNode;
    }
    else
    {
        (*tail)->next = newNode;
        newNode->prev = *tail;
        *tail = newNode;
    }
}
void displayList(struct Node* head)
{
    struct Node* current = head;
    while(current != NULL)
    {
        printf("%d ",current->studentID);
        current = current->next;
    }
    printf("\n");
}
void freeList(struct Node* head)
{
    struct Node* current = head;
    struct Node* next;
    while(current != NULL)
    {
        next = current->next;
    }
}
int main()
{
    struct Node* head = NULL;
    struct Node* tail = NULL;
    int n,studentID;
    scanf("%d",&n);
    for(int i=0; i<n; i++)
    {
        scanf("%d",&studentID);
        insertAtEnd(&head,&tail,studentID);
    }
```

```
    displayList(head);
    freeList(head);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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 data;
    struct Node *prev;
    struct Node *next;
};
struct Node *head = NULL;
struct Node *tail = NULL;

void insertEnd(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;
        tail = newNode;
    } else {
        newNode->prev = tail;
        tail->next = newNode;
        tail = newNode;
    }
}
void deleteNode(int position, int n) {
    if (position < 1 || position > n) {
        printf("Invalid position. Try again.\n");
        return;
    }

    struct Node *current = head;
    if (position == 1) {
        head = head->next;
        if (head != NULL) {
            head->prev = NULL;
        } else {
            tail = NULL;
        }
        free(current);
        return;
    }

    for (int i = 1; i < position; i++) {
```

```
        current = current->next;
    }

    if (current == tail) {
        tail = current->prev;
        tail->next = NULL;
    } else {
        current->prev->next = current->next;
        current->next->prev = current->prev;
    }
    free(current);
}

void displayList() {
    struct Node *current = head;
    int nodeNum = 1;
    while (current != NULL) {
        printf(" node %d : %d\n", nodeNum, current->data);
        current = current->next;
        nodeNum++;
    }
}

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

    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        insertEnd(data);
    }

    printf("Data entered in the list:\n");
    displayList();

    scanf("%d", &position);

    if (position > n || position <1) {
        printf("Invalid position. Try again.\n");
    } else {
        deleteNode(position, n);
```

```
        printf("\n After deletion the new list:\n");
        displayList();
    }

return 0;
}
```

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

Rajalakshmi Engineering College

Name: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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 data;
    struct Node *prev;
    struct Node *next;
};
struct Node *head = NULL;
struct Node *tail = NULL;

void insertEnd(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;
        tail = newNode;
    } else {
        newNode->prev = tail;
        tail->next = newNode;
        tail = newNode;
    }
}
void deleteNode(int position, int n) {
    if (position < 1 || position > n) {
        printf("Invalid position. Try again.\n");
        return;
    }

    struct Node *current = head;
    if (position == 1) {
        head = head->next;
        if (head != NULL) {
            head->prev = NULL;
        } else {
            tail = NULL;
        }
        free(current);
        return;
    }

    for (int i = 1; i < position; i++) {
```

```
        current = current->next;
    }

    if (current == tail) {
        tail = current->prev;
        tail->next = NULL;
    } else {
        current->prev->next = current->next;
        current->next->prev = current->prev;
    }
    free(current);
}

void displayList() {
    struct Node *current = head;
    int nodeNum = 1;
    while (current != NULL) {
        printf(" node %d : %d\n", nodeNum, current->data);
        current = current->next;
        nodeNum++;
    }
}

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

    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        insertEnd(data);
    }

    printf("Data entered in the list:\n");
    displayList();

    scanf("%d", &position);

    if (position > n || position <1) {
        printf("Invalid position. Try again.\n");
    } else {
        deleteNode(position, n);
```

```
        printf("\n After deletion the new list:\n");
        displayList();
    }

return 0;
}
```

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

Rajalakshmi Engineering College

Name: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

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

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

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
   int data;
   struct Node* prev;
   struct Node* next;
};
struct Node* createNode(int data)
{
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   if(newNode == NULL)
   {
      printf("Memory allocation failed\n");
      exit(1);
   }
   newNode->data = data;
   newNode->prev = NULL;
   newNode->next = NULL;
   return 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 displayList(struct Node* head)
{
    if(head == NULL)
    {
        printf("List is empty\n");
        return;
    }
    struct Node* temp = head;
    while(temp != NULL)
    {
        printf("%d ",temp->data);
        temp = temp->next;
    }
    printf("\n");
}
struct Node* rotateClockwise(struct Node* head,int k,int n)
{
    if(head == NULL || k == 0 || k%n == 0)
    {
        return head;
    }
    k = k % n;
    struct Node* tail = head;
    while(tail->next != NULL)
    {
        tail = tail->next;
    }
    for(int i=0; i<k; i++)
    {
        struct Node* temp = tail;
```

```
        tail = tail->prev;
        temp->next = head;
        head->prev = temp;
        head = temp;
        head->prev = NULL;
        tail->next = NULL;
    }
    return head;
}
int main()
{
    int n,data,k;
    struct Node* head = NULL;
    scanf("%d",&n);
    for(int i=0; i<n; i++)
    {
        scanf("%d",&data);
        insertAtEnd(&head,data);
    }
    scanf("%d",&k);
    head = rotateClockwise(head,k,n);
    displayList(head);
    struct Node* temp;
    while(head != NULL)
    {
        temp = head;
        head = head->next;
        free(temp);
    }
    return 0;
}
```

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

2. 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

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};
struct Node* createNode(int data)
```

```
{
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = 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 displayList(struct Node* head)
{
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
struct Node* deleteNodeFromBeginning(struct Node* head, int position)
{
    if (head == NULL)
    {
        printf("List is empty, nothing to delete\n");
        return NULL;
    }
```

```
    if (position <= 0)
    {
        printf("Invalid position. Position should be >= 1\n");
        return head;
    }
    struct Node* current = head;
    int count = 1;
    while (current != NULL && count < position)
    {
        current = current->next;
        count++;
    }
    if (current == NULL)
    {
        printf("Position is out of bounds.  List size is %d\n", count-1);
        return head;
    }
    if (current == head)
    {
        head = head->next;
        if (head != NULL)
        {
            head->prev = NULL;
        }
        free(current);
        return head;
    }
    else
    {
        current->prev->next = current->next;
        if (current->next != NULL)
        {
            current->next->prev = current->prev;
        }
        free(current);
        return head;
    }
}
int main()
{
    int n, data, x;
```

```
    struct Node* head = NULL;
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &data);
        insertAtBeginning(&head, data);
    }
    displayList(head);
    scanf("%d", &x);
    head = deleteNodeFromBeginning(head, x);
    displayList(head);
    struct Node* temp;
    while (head != NULL)
    {
        temp = head;
        head = head->next;
        free(temp);
    }
    return 0;
}
```

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

3. 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

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
   int data;
   struct Node* prev;
   struct Node* next;
};
struct Node* createNode(int data)
{
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   if(newNode == NULL)
   {
      printf("Memory allocation failed\n");
      exit(1);
   }
   newNode->data = data;
   newNode->prev = NULL;
   newNode->next = NULL;
   return 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 displayList(struct Node* head)
{
   if(head == NULL)
   {
      printf("List is empty\n");
      return;
   }
   struct Node* temp = head;
   while(temp != NULL)
   {
      printf("%d ",temp->data);
      temp = temp->next;
   }
   printf("\n");
}
void findMiddle(struct Node* head)
{
   if(head == NULL)
   {
      printf("List is empty\n");
      return;
   }
   struct Node* slow = head;
   struct Node* fast = head;
   int count = 0;
   while(fast != NULL && fast->next != NULL)
```

```
    {
        slow = slow->next;
        fast = fast->next->next;
        count++;
    }
    if(fast == NULL)
    {
        printf("%d %d\n",slow->prev->data,slow->data);
    }
    else
    {
        printf("%d\n",slow->data);
    }
}
int main()
{
    int n,data;
    struct Node* head = NULL;
    scanf("%d",&n);
    for(int i=0; i<n; i++)
    {
        scanf("%d",&data);
        insertAtEnd(&head,data);
    }
    displayList(head);
    findMiddle(head);
    struct Node* temp;
    while(head != NULL)
    {
        temp = head;
        head = head->next;
        free(temp);
    }
    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

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};
struct Node* createNode(int data)
{
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (newNode == NULL) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = 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 insertAtPosition(struct Node** head, int data, int position)
{
    if (position <= 0)
    {
        printf("Invalid position. Please enter a positive integer.\n");
        return;
    }
```

```
struct Node* newNode = createNode(data);
if (*head == NULL)
{
   if (position == 1)
   {
      *head = newNode;
   } else
   {
      printf("List is empty. Cannot insert at position %d.\n", position);
      free(newNode); //prevent memory leak
   }
   return;
}
if (position == 1)
{
   newNode->next = *head;
   (*head)->prev = newNode;
   *head = newNode;
   return;
}
struct Node* current = *head;
int count = 1;
while (current != NULL && count < position - 1)
{
   current = current->next;
   count++;
}

if (current == NULL)
{
   printf("Position is out of bounds.  The list has %d elements.\n", count);
   free(newNode);
   return;
}

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

```
}
void displayList(struct Node* head)
{
    if (head == NULL)
    {
        printf("List is empty\n");
        return;
    }
    struct Node* temp = head;
    while (temp != NULL)
    {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

int main()
{
    int n, data, position;
    struct Node* head = NULL;
    scanf("%d", &n);
    for (int i = 0; i < n; i++)
    {
        scanf("%d", &data);
        insertAtBeginning(&head, data);
    }
    displayList(head);
    scanf("%d", &position);
    scanf("%d", &data);
    insertAtPosition(&head, data, position);
    displayList(head);
    struct Node* temp;
    while (head != NULL)
    {
        temp = head;
        head = head->next;
        free(temp);
    }

return 0;
}
```

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

5. 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

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

```
#include<stdlib.h>
struct Node
{
   int data;
   struct Node* prev;
   struct Node* next;
};
struct Node* createNode(int data)
{
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   if(newNode == NULL)
   {
      printf("Memory allocation failed\n");
      exit(1);
   }
   newNode->data = data;
   newNode->prev = NULL;
   newNode->next = NULL;
   return 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 displayList(struct Node* head)
{
   struct Node* temp = head;
   while(temp != NULL)
   {
      printf("%d ",temp->data);
```

```
        temp = temp->next;
    }
    printf("\n");
}
int isPalindrome(struct Node* head)
{
    if(head == NULL || head->next == NULL)
    {
        return 1;
    }
    struct Node* front = head;
    struct Node* rear = head;
    while(rear->next != NULL)
    {
        rear = rear->next;
    }
    while(front != rear && front->prev != rear)
    {
        if(front->data != rear->data)
        {
            return 0;
        }
        front = front->next;
        rear = rear->prev;
    }
    return 1;
}
int main()
{
    int n,data;
    struct Node* head = NULL;
    scanf("%d",&n);
    for(int i=0; i<n; i++)
    {
        scanf("%d",&data);
        insertAtEnd(&head,data);
    }
    displayList(head);
    if(isPalindrome(head))
    {
        printf("The doubly linked list is a palindrome\n");
    }
```

```
    else
    {
        printf("The doubly linked list is not a palindrome\n");
    }
    struct Node* temp;
    while(head != NULL)
    {
        temp = head;
        head = head->next;
        free(temp);
    }
    return 0;
}
```

Status : Correct *Marks : 10/10*