Rajalakshmi Engineering College

Name: Mukesh kumar
Email: 241901058@rajalakshmi.edu.in
Roll no: 241901058
Phone: 8072542782
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_MCQ_Updated

Attempt : 1
Total Mark : 20
Marks Obtained : 15

Section 1 : MCQ

1. What does the following code snippet do?

```
struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
newNode->data = value;
newNode->next = NULL;
newNode->prev = NULL;
```

Answer

Creates a new node and initializes its data to 'value'

Status : Correct ***Marks : 1/1***

2. Consider the following function that refers to the head of a Doubly Linked List as the parameter. Assume that a node of a doubly linked list has the previous pointer as prev and the next pointer as next.

Assume that the reference of the head of the following doubly linked list is passed to the below function 1 <--> 2 <--> 3 <--> 4 <--> 5 <-->6. What should be the modified linked list after the function call?

```
Procedure fun(head_ref: Pointer to Pointer of node)
   temp = NULL
   current = *head_ref

   While current is not NULL
      temp = current->prev
      current->prev = current->next
      current->next = temp
      current = current->prev
   End While

   If temp is not NULL
      *head_ref = temp->prev
   End If
End Procedure
```

Answer

6 <--> 5 <--> 4 <--> 3 <--> 1 <--> 2.

Status : Wrong ***Marks : 0/1***

3. What is a memory-efficient double-linked list?

Answer

An auxiliary singly linked list acts as a helper list to traverse through the doubly linked list

Status : Wrong ***Marks : 0/1***

4. How do you delete a node from the middle of a doubly linked list?

Answer

All of the mentioned options

Status : Correct ***Marks : 1/1***

5. How many pointers does a node in a doubly linked list have?

Answer

3

Status : Wrong ***Marks : 0/1***

6. Which code snippet correctly deletes a node with a given value from a doubly linked list?

```
void deleteNode(Node** head_ref, Node* del_node) {
    if (*head_ref == NULL || del_node == NULL) {
        return;
    }
    if (*head_ref == del_node) {
        *head_ref = del_node->next;
    }
    if (del_node->next != NULL) {
        del_node->next->prev = del_node->prev;
    }
    if (del_node->prev != NULL) {
        del_node->prev->next = del_node->next;
    }
    free(del_node);
}
```

Answer

Deletes the node at a given position in a doubly linked list.

Status : Wrong ***Marks : 0/1***

7. How do you reverse a doubly linked list?

Answer

By changing the next pointer of each node to the previous node

Status : Wrong ***Marks : 0/1***

8. What will be the effect of setting the prev pointer of a node to NULL in a doubly linked list?

Answer

The node will become the new head

Status : Correct ***Marks : 1/1***

9. Which of the following is false about a doubly linked list?

Answer

Implementing a doubly linked list is easier than singly linked list

Status : Correct ***Marks : 1/1***

10. What is the main advantage of a two-way linked list over a one-way linked list?

Answer

Two-way linked lists allow for traversal in both directions.

Status : Correct ***Marks : 1/1***

11. Which of the following statements correctly creates a new node for a doubly linked list?

Answer

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

Status : Correct ***Marks : 1/1***

12. What will be the output of the following code?

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

```
#include <stdlib.h>

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

int main() {
    struct Node* head = NULL;
    struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
    temp->data = 2;
    temp->next = NULL;
    temp->prev = NULL;
    head = temp;
    printf("%d\n", head->data);
    free(temp);
    return 0;
}
```

Answer

2

Status : Correct ***Marks : 1/1***

13. Which of the following is true about the last node in a doubly linked list?

Answer

Its next pointer is NULL

Status : Correct ***Marks : 1/1***

14. What happens if we insert a node at the beginning of a doubly linked list?

Answer

The previous pointer of the new node is NULL

Status : Correct **Marks : 1/1**

15. What is the correct way to add a node at the beginning of a doubly linked list?

Answer

void addFirst(int data){ Node* newNode = new Node(data); newNode->next = head; if (head != NULL) { head->prev = newNode; } head = newNode; }

Status : Correct **Marks : 1/1**

16. Where Fwd and Bwd represent forward and backward links to the adjacent elements of the list. Which of the following segments of code deletes the node pointed to by X from the doubly linked list, if it is assumed that X points to neither the first nor the last node of the list?

A doubly linked list is declared as

```
struct Node {
    int Value;
    struct Node *Fwd;
    struct Node *Bwd;
);
```

Answer

X->Bwd->Fwd = X->Fwd; X->Fwd->Bwd = X->Bwd;

Status : Correct **Marks : 1/1**

17. What will be the output of the following program?

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

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

```
    struct Node* prev;
};

int main() {
    struct Node* head = NULL;
    struct Node* tail = NULL;
    for (int i = 0; i < 5; i++) {
        struct Node* temp = (struct Node*)malloc(sizeof(struct Node));
        temp->data = i + 1;
        temp->prev = tail;
        temp->next = NULL;
        if (tail != NULL) {
            tail->next = temp;
        } else {
            head = temp;
        }
        tail = temp;
    }
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    return 0;
}
```

Answer

1 2 3 4 5

Status : Correct ***Marks : 1/1***

18. Which of the following information is stored in a doubly-linked list's nodes?

Answer

All of the mentioned options

Status : Correct ***Marks : 1/1***

19. Which pointer helps in traversing a doubly linked list in reverse order?

Answer

prev

Status : Correct ***Marks : 1/1***

20. Consider the provided pseudo code. How can you initialize an empty two-way linked list?

```
Define Structure Node
   data: Integer
   prev: Pointer to Node
   next: Pointer to Node
End Define

Define Structure TwoWayLinkedList
   head: Pointer to Node
   tail: Pointer to Node
End Define
```

Answer

struct TwoWayLinkedList* list = malloc(sizeof(struct TwoWayLinkedList)); list->head = NULL; list->tail = NULL;

Status : Correct ***Marks : 1/1***

Rajalakshmi Engineering College

Name: Mukesh kumar
Email: 241901058@rajalakshmi.edu.in
Roll no: 241901058
Phone: 8072542782
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) {
   //type your code here
   struct Node*newnode=(struct Node*)malloc(sizeof(struct Node));
   newnode->item=item;
   newnode->next=NULL;
   newnode->prev=NULL;
   if(*head==NULL){
      *head=newnode;
       newnode->prev=NULL;
```

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

    }
    temp->next=newnode;
    newnode->prev=temp;
    return;

}
void displayForward(struct Node* head) {
    struct Node*temp=head;
    while(temp){
        printf("%c ",temp->item);
        temp=temp->next;
    }
    printf("\n");
    return;
}

void displayBackward(struct Node* tail) {
    //type your code here
    while(tail){
        printf("%s ",tail);
        tail=tail->prev;
    }
    printf("\n");
    return;
}

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

    Node*current=head;
    Node*next;

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

```
    }
}
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: Mukesh kumar
Email: 241901058@rajalakshmi.edu.in
Roll no: 241901058
Phone: 8072542782
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>
typedef struct Node{
    int data;
    struct Node *next;
    struct Node*prev;

} Node;
Node* createnode(int data){

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


void append(Node**head_ref,int newdata){
    Node*newnode=createnode(newdata);
    Node*last=*head_ref;
    if(*head_ref==NULL){
        *head_ref=newnode;
        return;
```

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

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

int findmax(Node*head){
    if(head==NULL){
        return -1;

    }
    int max=head->data;
    Node*temp=head->next;
    while(temp!=NULL){
        if(temp->data>max)
            max=temp->data;

        temp=temp->next;
    }
    return max;
}


int main(){
    int n,id;
    Node*head=NULL;
    scanf("%d",&n);
    if(n==0){
        printf("Empty list!\n");
        return 0;
    }
    for (int i=0;i<n;i++){
        scanf("%d",&id);
        append(&head,id);
    }
    int maxid=findmax(head);
    if(maxid==-1){
        printf("Empty list!\n");
    }else{
```

```
        printf("%d\n",maxid);

    }
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Mukesh kumar
Email: 241901058@rajalakshmi.edu.in
Roll no: 241901058
Phone: 8072542782
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() {
    printf("Node Inserted\n");
    struct node*temp=start;
    while(temp){
        printf("%d ",temp->info);
        temp=temp->next;
    }
    printf("\n");
```

```
    return;

    //type your code here
}

void insertAtFront(int data) {
    //type your code here
    struct node*newnode=(struct node*)malloc(sizeof(struct node));
    newnode->info=data;
    newnode->next=NULL;
    newnode->prev=NULL;
    if(start==NULL){
        start=newnode;
        return;
    }
    else{
        start->prev=newnode;
        newnode->next=start;
        start=newnode;
    }
    return;
}

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: Mukesh kumar
Email: 241901058@rajalakshmi.edu.in
Roll no: 241901058
Phone: 8072542782
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

```
#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->next=NULL;
   newnode->prev-NULL;
   return newnode;

}

void printlist(node*head){
   node*temp=head;
   head=temp;
   while(temp!=NULL){
      printf("%d ",temp->data);
      temp=temp->next;
   }
```

```
}

int main(){
    int n;
    scanf("%d",&n);
    node*head=NULL;
    node*tail=NULL;
    for(int i=0;i<n;i++){
        int id;
        scanf("%d",&id);
        node*newnode=createnode(id);
        if(head==NULL){
            head=newnode;
            tail=newnode;
        }
        else{
            tail->next=newnode;
            newnode->prev=tail;
            tail=newnode;
        }
    }
    printlist(head);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Mukesh kumar
Email: 241901058@rajalakshmi.edu.in
Roll no: 241901058
Phone: 8072542782
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>
```

```
typedef struct node{
    int data;
    struct node*next;
    struct node*prev;

}node;

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

void display(node*head){
    node*temp=head;
    int nodecount=1;
    while(temp!=NULL){
        printf("node %d : %d\n",nodecount++,temp->data);
        temp=temp->next;
    }
}

int getlength(node*head){
    int count=0;
    node*temp=head;
    while(temp!=NULL){
        count++;
        temp=temp->next;
    }
    return count;
}

node*deleteatpos(node*head,int pos){
    if(pos<=0 || head==NULL)
    return head;
    node*temp=head;
    int i=1;
    while(temp!=NULL && i<pos){
        temp=temp->next;
        i++;
```

```
    }
    if(temp==NULL)
    return head;
    if(temp->prev!=NULL)
    temp->prev->next=temp->next;
    else{
        head=temp->next;
    }
    if(temp->next!=NULL){
        temp->next->prev=temp->prev;
    }
    free(temp);
    return head;
}



int main(){
    int n,pos;
    scanf("%d",&n);
    node*head=NULL;
    node*tail=NULL;
    for(int i=0;i<n;i++){
        int val;
        scanf("%d",&val);
        node*newnode=createnode(val);
        if(head==NULL){
            head=tail=newnode;

        }
        else{
            tail->next=newnode;
            newnode->prev=tail;
            tail=newnode;
        }
    }
    scanf("%d",&pos);
    printf("Data entered in the list:\n");
    display(head);
    if(pos<1 || pos>getlength(head)){
        printf("Invalid position. Try again.\n");
    }
```

```
    else{
        head=deleteatpos(head,pos);
        printf("After deletion the new list:\n");
        display(head);
    }
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Mukesh kumar
Email: 241901058@rajalakshmi.edu.in
Roll no: 241901058
Phone: 8072542782
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 : 2
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

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

Input Format

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

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

Output Format

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

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

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50

Output: 10 20 30 40 50
30

Answer

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

struct Node {
    int data;
    Node* previous;
    Node* next;
};

struct LinkedList {
    int size;
    Node* head;
    Node* tail;
};

void initializeLinkedList(LinkedList* list) {
    list->size = 0;
    list->head = list->tail = nullptr;
}

void addNode(LinkedList* list, int data) {
    Node* newNode = new Node;
    newNode->data = data;
    if (list->head == nullptr) {
        list->head = list->tail = newNode;
        list->head->previous = nullptr;
```

```
        list->tail->next = nullptr;
    } else {
        list->tail->next = newNode;
        newNode->previous = list->tail;
        list->tail = newNode;
        list->tail->next = nullptr;
    }
    list->size++;
}

void middle(LinkedList* list) {
    Node* temp = list->head;
    int c = 0;
    while (temp != nullptr) {
        c++;
        temp = temp->next;
    }

    temp = list->head;
    int p = 1;
    if (c % 2 == 0) {
        int mid = c / 2;
        while (temp != nullptr) {
            if (p == mid)
                break;
            p++;
            temp = temp->next;
        }
        cout << temp->data << " " << temp->next->data << endl;
    } else {
        int mid = (c + 1) / 2;
        while (temp != nullptr) {
            if (p == mid)
                break;
            p++;
            temp = temp->next;
        }
        cout << temp->data << endl;
    }
}

void display(LinkedList* list) {
```

```
    Node* current = list->head;
    if (list->head == nullptr) {
        cout << "List is empty" << endl;
        return;
    }
    while (current != nullptr) {
        cout << current->data << " ";
        current = current->next;
    }
    cout << endl;
}

int main() {
    LinkedList list;
    int n, ele;
    cin >> n;
    initializeLinkedList(&list);

    for (int i = 0; i < n; i++) {
        cin >> ele;
        addNode(&list, ele);
    }

    display(&list);
    middle(&list);

    return 0;
}
```

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

2. 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>
#include <stdlib.h>

// Define a node of the doubly linked list
typedef struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
} Node;

// Insert a new node at the end of the list
void insertAtEnd(Node** head, int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
```

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

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

// Display the list in forward order
void displayForward(Node* head) {
    printf("List in original order:\n");
    Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// Display the list in reverse order
void displayReverse(Node* head) {
    printf("List in reverse order:\n");
    Node* temp = head;

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

    // Traverse backward
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->prev;
    }
    printf("\n");
```

```
}
int main() {
    Node* head = NULL;
    int n, data;

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

    // Read each element and insert at end
    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        insertAtEnd(&head, data);
    }

    // Display list
    displayForward(head);
    displayReverse(head);

    return 0;
}
```

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

3. Problem Statement

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

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

Input Format

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

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

list elements.

Output Format

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

Refer to the sample output for formatting specifications.

Sample Test Case

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

Answer

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.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;
}


struct Node* deleteNode(struct Node* head_ref, struct Node* del) {
    if (head_ref == NULL || del == NULL)
        return NULL;

    if (head_ref == del)
        head_ref = del->next;
```

```
    if (del->next != NULL)
        del->next->prev = del->prev;

    if (del->prev != NULL)
        del->prev->next = del->next;

    free(del);
    return head_ref;
}

bool exists(int* array, int size, int value) {
    for (int i = 0; i < size; i++) {
        if (array[i] == value)
            return true;
    }
    return false;
}

struct Node* removeDuplicates(struct Node* head_ref) {
    if (head_ref == NULL)
        return NULL;

    int* us = (int*)malloc(sizeof(int) * 1000);
    int us_size = 0;

    struct Node* current = head_ref;
    struct Node* next;

    while (current != NULL) {
        if (exists(us, us_size, current->data)) {
            next = current->next;
            head_ref = deleteNode(head_ref, current);
            current = next;
        } else {
            us[us_size++] = current->data;
            current = current->next;
        }
    }

    free(us);
    return head_ref;
}
```

```
struct Node* push(struct Node* head_ref, int new_data) {
    struct Node* new_node = createNode(new_data);
    new_node->next = head_ref;

    if (head_ref != NULL)
        head_ref->prev = new_node;

    head_ref = new_node;
    return head_ref;
}

void printList(struct Node* head) {
    if (head == NULL) {
        printf("Doubly Linked list empty\n");
        return;
    }

    while (head != NULL) {
        printf("%d ", head->data);
        head = head->next;
    }
}

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

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

    head = removeDuplicates(head);
    printList(head);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Mukesh kumar
Email: 241901058@rajalakshmi.edu.in
Roll no: 241901058
Phone: 8072542782
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 : 2
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>

// Define the structure for a node
typedef struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
} Node;

// Function to insert a node at the end
void insertAtEnd(Node** head, int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;

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

```
        *head = newNode;
        return;
    }

    Node* temp = *head;
    while (temp->next != NULL)
        temp = temp->next;

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

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

// Function to print the middle element(s)
void printMiddle(Node* head, int n) {
    int mid = n / 2;
    Node* temp = head;

    for (int i = 0; i < mid; i++) {
        temp = temp->next;
    }

    if (n % 2 == 0) {
        // Even number of elements: print mid-1 and mid
        printf("%d %d\n", temp->prev->data, temp->data);
    } else {
        // Odd number of elements: print the middle
        printf("%d\n", temp->data);
    }
}

// Main function
int main() {
```

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

    Node* head = NULL;

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

    printList(head);
    printMiddle(head, n);

    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>

// Define the structure for a node
typedef struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
} Node;

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

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

    *head = newNode;
}

// Function to delete a node at a given position
void deleteAtPosition(Node** head, int pos) {
    if (*head == NULL) return;

    Node* temp = *head;

    // If deleting the head node
    if (pos == 1) {
        *head = temp->next;
        if (*head != NULL)
            (*head)->prev = NULL;
        free(temp);
        return;
    }

    // Traverse to the (pos - 1)th node
    for (int i = 1; temp != NULL && i < pos; i++) {
        temp = temp->next;
    }

    // If position is beyond list length
    if (temp == NULL)
        return;

    if (temp->next != NULL)
        temp->next->prev = temp->prev;
    if (temp->prev != NULL)
        temp->prev->next = temp->next;

    free(temp);
}

// Function to print the list
void printList(Node* head) {
    Node* temp = head;
```

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

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

    Node* head = NULL;
    for (int i = 0; i < N; i++) {
        int val;
        scanf("%d", &val);
        insertAtFront(&head, val);
    }

    scanf("%d", &x);

    printList(head);
    deleteAtPosition(&head, x);
    printList(head);

    return 0;
}
```

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

3. 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>

// Define the structure of a node
typedef struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
} Node;

// Function to insert at end
void insertAtEnd(Node** head, int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;

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

```
        return;
    }

    Node* temp = *head;
    while (temp->next != NULL)
        temp = temp->next;

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

// Function to rotate the list clockwise by k positions
void rotateClockwise(Node** head, int k) {
    if (*head == NULL || k == 0)
        return;

    Node* tail = *head;
    int count = 1;

    // Traverse to the last node
    while (tail->next != NULL) {
        tail = tail->next;
        count++;
    }

    k = k % count; // In case k > count
    if (k == 0) return;

    // Traverse to the (count - k)th node
    Node* newTail = *head;
    for (int i = 0; i < count - k - 1; i++) {
        newTail = newTail->next;
    }

    Node* newHead = newTail->next;

    // Update links
    newTail->next = NULL;
    newHead->prev = NULL;
    tail->next = *head;
    (*head)->prev = tail;
```

```
    *head = newHead;
}

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

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

    Node* head = NULL;

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

    scanf("%d", &k);

    rotateClockwise(&head, k);
    printList(head);

    return 0;
}
```

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

4. 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>
#include <stdbool.h>

// Define the node structure
typedef struct Node {
    int data;
    struct Node* prev;
```

```
    struct Node* next;
} Node;

// Function to insert at the end of the list
void insertAtEnd(Node** head, int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = NULL;

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

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

// Function to check if the list is a palindrome
bool isPalindrome(Node* head) {
    if (head == NULL) return true;

    Node* left = head;
    Node* right = head;

    // Move right to the end
    while (right->next != NULL) {
        right = right->next;
```

```
    }

    // Compare from both ends
    while (left != NULL && right != NULL && left != right && left->prev != right) {
        if (left->data != right->data) {
            return false;
        }
        left = left->next;
        right = right->prev;
    }

    return true;
}

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

    Node* head = NULL;

    for (int i = 0; i < N; i++) {
        int value;
        scanf("%d", &value);
        insertAtEnd(&head, value);
    }

    printList(head);

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

    return 0;
}
```

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

5. 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>

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

// Insert at the front of the list
void insertAtFront(Node** head, int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->prev = NULL;
    newNode->next = *head;

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

// Insert at a given position (1-based index)
void insertAtPosition(Node** head, int position, int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;

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

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

```
    if (temp == NULL) return; // Invalid position safety

    newNode->next = temp->next;
    newNode->prev = temp;

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

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

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

    Node* head = NULL;

    for (int i = 0; i < N; i++) {
        int value;
        scanf("%d", &value);
        insertAtFront(&head, value);
    }

    printList(head);

    int position, data;
    scanf("%d %d", &position, &data);

    insertAtPosition(&head, position, data);

    printList(head);
```

```
    return 0;
}
```

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