Rajalakshmi Engineering College

Name: Hemaprakash KL
Email: 241901036@rajalakshmi.edu.in
Roll no: 241901036
Phone: 7010799637
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

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>

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

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

// Function to append a node to the list
void append(Node** head, int data) {
```

```
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
    while (temp->next) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->prev = temp;
}

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

// Function to check if the list is a palindrome
int isPalindrome(Node* head) {
    if (head == NULL) return 1; // Empty list is a palindrome

    // Find the last node
    Node* last = head;
    while (last->next) {
        last = last->next;
    }

    // Compare from both ends
    while (head != last && head->prev != last) {
        if (head->data != last->data) {
            return 0;
        }
        head = head->next;
        last = last->prev;
    }
```

```
    return 1;
}

int main() {
    int N;
    Node* head = NULL;

    // Read input
    scanf("%d", &N);
    int elements[N];
    for (int i = 0; i < N; i++) {
        scanf("%d", &elements[i]);
        append(&head, elements[i]);
    }

    // Display the list
    display(head);

    // Check if the list is a palindrome
    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**

2. 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>

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

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

```
// Function to append a node to the list
void append(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
    while (temp->next) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->prev = temp;
}

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

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

    Node* last = *head;
    while (last->next) {
        last = last->next;
    }

    Node* newLast = *head;
    for (int i = 0; i < n - k - 1; i++) {
        newLast = newLast->next;
    }

    Node* newHead = newLast->next;
    newHead->prev = NULL;
    newLast->next = NULL;
```

```
    last->next = *head;
    (*head)->prev = last;

    *head = newHead;
}

// Main function to take input and process rotation
int main() {
    int n, k;
    Node* head = NULL;

    // Read input
    scanf("%d", &n);
    int elements[n];

    for (int i = 0; i < n; i++) {
        scanf("%d", &elements[i]);
        append(&head, elements[i]);
    }

    scanf("%d", &k);

    // Rotate the list
    rotateClockwise(&head, k, n);

    // Display the rotated list
    display(head);

    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>

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

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

// Function to append a node to the list
void append(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
    while (temp->next) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->prev = temp;
}

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

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

    // Move to the middle node
    for (int i = 0; i < mid; i++) {
```

```
        temp = temp->next;
    }

    // If n is odd, print one middle element
    if (n % 2 != 0) {
        printf("%d\n", temp->data);
    }
    // If n is even, print two middle elements
    else {
        printf("%d %d\n", temp->prev->data, temp->data);
    }
}

int main() {
    int n;
    Node* head = NULL;

    // Read input
    scanf("%d", &n);
    int elements[n];

    for (int i = 0; i < n; i++) {
        scanf("%d", &elements[i]);
        append(&head, elements[i]);
    }

    // Display the list
    display(head);

    // Print the middle element(s)
    printMiddle(head, n);

    return 0;
}
```

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

4. 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>
```

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

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

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

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

// Function to delete a node at a given position
void deleteAtPosition(Node** head, int pos) {
    if (*head == NULL || pos <= 0) return;

    Node* temp = *head;
    int count = 1;
```

```
    // Traverse to the position
    while (temp && count < pos) {
        temp = temp->next;
        count++;
    }

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

    // Adjust pointers
    if (temp->prev)
        temp->prev->next = temp->next;
    if (temp->next)
        temp->next->prev = temp->prev;

    // Update head if needed
    if (temp == *head)
        *head = temp->next;

    free(temp);
}

int main() {
    int N, X;
    Node* head = NULL;

    // Read input
    scanf("%d", &N);
    int elements[N];

    for (int i = 0; i < N; i++) {
        scanf("%d", &elements[i]);
        appendFront(&head, elements[i]);
    }

    scanf("%d", &X);

    // Display the original list
    display(head);

    // Delete the node at position X
    deleteAtPosition(&head, X);
```

```
    // Display the updated list
    display(head);

    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

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

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

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

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

```
    *head = newNode;
}

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

// Function to insert a node at a specific position
void insertAtPosition(Node** head, int pos, int data) {
    Node* newNode = createNode(data);
    if (pos == 1) { // Insert at the front
        newNode->next = *head;
        if (*head) (*head)->prev = newNode;
        *head = newNode;
        return;
    }

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

    while (temp && count < pos - 1) {
        temp = temp->next;
        count++;
    }

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

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

int main() {
    int N, pos, data;
    Node* head = NULL;
```

```
    // Read input
    scanf("%d", &N);
    int elements[N];

    for (int i = 0; i < N; i++) {
        scanf("%d", &elements[i]);
        insertFront(&head, elements[i]);
    }

    scanf("%d", &pos);
    scanf("%d", &data);

    // Display the original list
    display(head);

    // Insert the new contact at the specified position
    insertAtPosition(&head, pos, data);

    // Display the updated list
    display(head);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Hemaprakash KL
Email: 241901036@rajalakshmi.edu.in
Roll no: 241901036
Phone: 7010799637
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

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

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

// Function to create a new node
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;
}

// Function to append a node to the doubly linked list
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;
}

// Function to display the list in original order
void displayOriginalOrder(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// Function to display the list in reverse order
void displayReverseOrder(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL && temp->next != NULL) {
        temp = temp->next;
    }
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->prev;
    }
    printf("\n");
}

// Main function
int main() {
    int n, i, element;
    struct Node* head = NULL;

    // Input number of elements
```

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

    // Input elements and append to the list
    for (i = 0; i < n; i++) {
        scanf("%d", &element);
        append(&head, element);
    }

    // Output
    printf("List in original order:\n");
    displayOriginalOrder(head);
    printf("List in reverse order:\n");
    displayReverseOrder(head);

    return 0;
}
```

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

2. Problem Statement

Ashiq is developing a ticketing system for a small amusement park. The park issues tickets to visitors in the order they arrive. However, due to a system change, the oldest ticket (first inserted) must be revoked instead of the last one.

To manage this, Ashiq decided to use a doubly linked list-based stack, where:

Pushing adds a new ticket to the top of the stack.Removing the first inserted ticket (removing from the bottom of the stack).Printing the remaining tickets from bottom to top.

Input Format

The first line consists of an integer n, representing the number of tickets issued.

The second line consists of n space-separated integers, each representing a ticket number in the order they were issued.

Output Format

The output prints space-separated integers, representing the remaining ticket numbers in the order from bottom to top.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 7
24 96 41 85 97 91 13

Output: 96 41 85 97 91 13

Answer

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

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

// Function to create a new node
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;
}

// Function to push a ticket onto the stack
void push(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    newNode->next = *head;
```

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

// Function to remove the oldest ticket (bottom of the stack)
void removeOldest(struct Node** head) {
    if (*head == NULL) return; // No tickets to remove
    struct Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    if (temp->prev != NULL) {
        temp->prev->next = NULL;
    } else {
        *head = NULL; // The list is now empty
    }
    free(temp);
}

// Function to print tickets from bottom to top
void printTickets(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL && temp->next != NULL) {
        temp = temp->next;
    }
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->prev;
    }
    printf("\n");
}

// Main function
int main() {
    int n, ticket;
    struct Node* head = NULL;

    // Input number of tickets
    scanf("%d", &n);

    // Input ticket numbers and push onto the stack
    for (int i = 0; i < n; i++) {
```

```
        scanf("%d", &ticket);
        push(&head, ticket);
    }

    // Remove the oldest ticket
    removeOldest(&head);

    // Print remaining tickets
    printTickets(head);

    return 0;
}
```

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

3. Problem Statement

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

Help her implement a program for the same.

Input Format

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

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

Output Format

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

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

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5

Output: 5 4 3 2 1
1 2 3 4 5

Answer

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

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

// Function to create a new node
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;
}

// Function to insert a node at the beginning
struct Node* insertAtBeginning(struct Node* head, int data) {
    struct Node* newNode = createNode(data);
    if (head != NULL) {
        newNode->next = head;
        head->prev = newNode;
    }
    return newNode; // New head
}

// Function to insert a node at the end
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;
}

// Function to print the list
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);

    int values[N]; // Array to store input values

    // Store input values in an array
    for (int i = 0; i < N; i++) {
        scanf("%d", &values[i]);
    }

    struct Node* head = NULL;

    // Insert at the beginning using stored values
    for (int i = 0; i < N; i++) {
        head = insertAtBeginning(head, values[i]);
    }

    // Print list after inserting at the beginning
    printList(head);
```

```
    struct Node* endHead = NULL;

    // Insert at the end using stored values
    for (int i = 0; i < N; i++) {
        insertAtEnd(&endHead, values[i]);
    }

    // Print list after inserting at the end
    printList(endHead);

    return 0;
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Hemaprakash KL
Email: 241901036@rajalakshmi.edu.in
Roll no: 241901036
Phone: 7010799637
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 : 4.5

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

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

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

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

void append(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
    while (temp->next) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->prev = temp;
}

void display(Node* head) {
    Node* temp = head;
    int pos = 1;
    printf("Data entered in the list:\n");
    while (temp) {
        printf(" node %d : %d\n", pos, temp->data);
        temp = temp->next;
        pos++;
    }
}

void deleteAtPosition(Node** head, int pos, int n) {
    if (pos <= 0 || pos > n) {
        printf("Invalid position. Try again.\n");
```

```
        return;
    }

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

    while (temp && count < pos) {
        temp = temp->next;
        count++;
    }

    if (temp == NULL) {
        printf("Invalid position. Try again.\n");
        return;
    }

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

    if (temp == *head)
        *head = temp->next;

    free(temp);

    printf("\nAfter deletion the new list:\n");
    display(*head);
}

int main() {
    int n, p;
    Node* head = NULL;

    scanf("%d", &n);
    int items[n];

    for (int i = 0; i < n; i++) {
        scanf("%d", &items[i]);
        append(&head, items[i]);
    }
```

```
    scanf("%d", &p);

    display(head);
    deleteAtPosition(&head, p, n);

    return 0;
}
```

Status : Partially correct ***Marks : 4.5/10***

Rajalakshmi Engineering College

Name: Hemaprakash KL
Email: 241901036@rajalakshmi.edu.in
Roll no: 241901036
Phone: 7010799637
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>

// Define the structure for a node in the doubly linked list
struct Node {
    int student_id;
    struct Node* prev;
    struct Node* next;
};

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

// Function to insert a student ID into the doubly linked list
void insert(struct Node** head, struct Node** tail, int student_id) {
    struct Node* newNode = createNode(student_id);
    if (*head == NULL) {
        // If the list is empty, set both head and tail to the new node
```

```
        *head = *tail = newNode;
    } else {
        // Insert at the end of the list
        (*tail)->next = newNode;
        newNode->prev = *tail;
        *tail = newNode;
    }
}

// Function to display the student IDs in the list
void display(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->student_id);
        current = current->next;
    }
    printf("\n");
}

// Main function to execute the program
int main() {
    int N;

    // Read the number of student IDs
    scanf("%d", &N);

    int student_id;
    struct Node* head = NULL;
    struct Node* tail = NULL;

    // Read the student IDs and insert them into the doubly linked list
    for (int i = 0; i < N; i++) {
        scanf("%d", &student_id);
        insert(&head, &tail, student_id);
    }

    // Display the student IDs in the order they were entered
    display(head);

    // Free the allocated memory for the linked list nodes
    struct Node* temp;
    while (head != NULL) {
```

```
        temp = head;
        head = head->next;
        free(temp);
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Hemaprakash KL
Email: 241901036@rajalakshmi.edu.in
Roll no: 241901036
Phone: 7010799637
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 : 0

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

-

Status : Skipped ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Hemaprakash KL
Email: 241901036@rajalakshmi.edu.in
Roll no: 241901036
Phone: 7010799637
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;
};
```

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

// Node structure for doubly linked list
struct Node {
    char data;
    Node* prev;
    Node* next;
};
```

```
// Function to create a new node
Node* createNode(char data) {
    Node* newNode = new Node();
    newNode->data = data;
    newNode->prev = nullptr;
    newNode->next = nullptr;
    return newNode;
}

// Function to insert item at the end
void insertEnd(Node*& head, Node*& tail, char data) {
    Node* newNode = createNode(data);
    if (tail == nullptr) {
        head = tail = newNode;
        return;
    }
    tail->next = newNode;
    newNode->prev = tail;
    tail = newNode;
}

// Function to insert item at the front
void insertFront(Node*& head, Node*& tail, char data) {
    Node* newNode = createNode(data);
    if (head == nullptr) {
        head = tail = newNode;
        return;
    }
    head->prev = newNode;
    newNode->next = head;
    head = newNode;
}

// Function to display playlist forward
void displayForward(Node* head) {
    cout << "Forward Playlist: ";
    Node* temp = head;
    while (temp) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
```

```
}

// Function to display playlist backward
void displayBackward(Node* tail) {
    cout << "Backward Playlist: ";
    Node* temp = tail;  // Start from the tail, correctly reversing the list
    while (temp) {
        cout << temp->data << " ";
        temp = temp->prev;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr;
    Node* tail = nullptr;
    char input;

    // Read input
    while (true) {
        cin >> input;
        if (input == '-') break;
        insertEnd(head, tail, input);
    }

    // Display forward list
    displayForward(head);

    // Display backward list directly using tail
    displayBackward(tail);

    return 0;
}


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: Hemaprakash KL
Email: 241901036@rajalakshmi.edu.in
Roll no: 241901036
Phone: 7010799637
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 : 0

Section 1 : MCQ

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

Answer

Status : Skipped ***Marks : 0/1***

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

Answer

-

Status : - ***Marks : 0/1***

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

Answer

-

Status : *-* ***Marks : 0/1***

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

Answer

-

Status : *-* ***Marks : 0/1***

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

Answer

-

Status : *-* ***Marks : 0/1***

6. 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

-

Status : *-* ***Marks : 0/1***

7. How do you reverse a doubly linked list?

Answer

-

Status : *-* ***Marks : 0/1***

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

Answer

-

Status : *-* ***Marks : 0/1***

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

Answer

-

Status : - ***Marks : 0/1***

10. 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

-

Status : - ***Marks : 0/1***

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

Answer

-

Status : - ***Marks : 0/1***

12. 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

-

Status : *-* ***Marks : 0/1***

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

Answer

-

Status : *-* ***Marks : 0/1***

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

Answer

-

Status : *-* ***Marks : 0/1***

15. 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

-

Status : - ***Marks : 0/1***

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

-

Status : - ***Marks : 0/1***

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

Answer

-

Status : - ***Marks : 0/1***

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

Answer

-

Status : - ***Marks : 0/1***

19. 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

-

Status : -

Marks : 0/1

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

-

Status : -

Marks : 0/1