Rajalakshmi Engineering College

Name: Priyan S
Email: 240701402@rajalakshmi.edu.in
Roll no: 240701402
Phone: 9150170939
Branch: REC
Department: I CSE FD
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 2_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. Problem Statement

Sam is learning about two-way linked lists. He came across a problem where he had to populate a two-way linked list and print the original as well as the reverse order of the list. Assist him with a suitable program.

Input Format

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

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

Output Format

The first line displays the message: "List in original order:"

The second line displays the elements of the doubly linked list in the original order.

The third line displays the message: "List in reverse order:"

The fourth line displays the elements of the doubly linked list in reverse order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5
Output: List in original order:
1 2 3 4 5
List in reverse order:
5 4 3 2 1

Answer

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

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, Node** tail, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
```

```
        *head = *tail = newNode;
    } else {
        (*tail)->next = newNode;
        newNode->prev = *tail;
        *tail = newNode;
    }
}

void printOriginalOrder(Node* head) {
    printf("List in original order:\n");
    while (head) {
        printf("%d ", head->data);
        head = head->next;
    }
    printf("\n");
}

void printReverseOrder(Node* tail) {
    printf("List in reverse order:\n");
    while (tail) {
        printf("%d ", tail->data);
        tail = tail->prev;
    }
    printf("\n");
}

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

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

    printOriginalOrder(head);
    printReverseOrder(tail);

    Node* temp;
    while (head) {
```

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

    return 0;
}
```

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

2. Problem Statement

You are required to implement a program that deals with a doubly linked list.

The program should allow users to perform the following operations:

Insertion at the End: Insert a node with a given integer data at the end of the doubly linked list.Insertion at a given Position: Insert a node with a given integer data at a specified position within the doubly linked list.Display the List: Display the elements of the doubly linked list.

Input Format

The first line of input consists of an integer n, representing the number of elements to be initially inserted into the doubly linked list.

The second line consists of n space-separated integers, denoting the elements to be inserted at the end.

The third line consists of integer m, representing the new element to be inserted.

The fourth line consists of an integer p, representing the position at which the new element should be inserted (1-based indexing).

Output Format

If p is valid, display the elements of the doubly linked list after performing the insertion at the specified position.

If p is invalid, display "Invalid position" in the first line and the second line prints the original list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 25 34 48 57
35
4

Output: 10 25 34 35 48 57

Answer

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

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, Node** tail, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = *tail = newNode;
    } else {
        (*tail)->next = newNode;
        newNode->prev = *tail;
        *tail = newNode;
    }
}
```

```
void insertAtPosition(Node** head, Node** tail, int data, int pos, int n) {
    if (pos < 1 || pos > n + 1) {
        printf("Invalid position\n");
        return;
    }

    Node* newNode = createNode(data);
    if (pos == 1) {
        newNode->next = *head;
        if (*head) (*head)->prev = newNode;
        *head = newNode;
        if (*tail == NULL) *tail = newNode;
    } else {
        Node* temp = *head;
        for (int i = 1; i < pos - 1; i++) {
            if (temp) temp = temp->next;
        }
        if (temp) {
            newNode->next = temp->next;
            newNode->prev = temp;
            if (temp->next) temp->next->prev = newNode;
            temp->next = newNode;
            if (newNode->next == NULL) *tail = newNode;
        }
    }
}

void printList(Node* head) {
    while (head) {
        printf("%d ", head->data);
        head = head->next;
    }
    printf("\n");
}

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

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

```
        scanf("%d", &data);
        append(&head, &tail, data);
    }

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

    if (p > n + 1) {
        printf("Invalid position\n");
        printList(head);
    } else {
        insertAtPosition(&head, &tail, m, p, n);
        printList(head);
    }

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

    return 0;
}
```

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

3. Problem Statement

Krishna needs to create a doubly linked list to store and display a sequence of integers. Your task is to help write a program to read a list of integers from input, store them in a doubly linked list, and then display the list.

Input Format

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

The second line of input consists of n space-separated integers.

Output Format

The output prints a single line displaying the integers in the order they were added to the doubly linked list, separated by spaces.

If nothing is added (i.e., the list is empty), it will display "List is empty".

Refer to the sample output for the formatting specifications.

Sample Test Case

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

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, Node** tail, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = *tail = newNode;
    } else {
        (*tail)->next = newNode;
```

```
        newNode->prev = *tail;
        *tail = newNode;
    }
}

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

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

    scanf("%d", &n);
    if (n == 0) {
        printf("List is empty\n");
        return 0;
    }

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

    printList(head);

    Node* temp;
    while (head) {
        temp = head;
        head = head->next;
        free(temp);
    }
```

```
    return 0;
}
```

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