Rajalakshmi Engineering College

Name: Ravi Sankar
Email: 240701424@rajalakshmi.edu.in
Roll no: 240701424
Phone: 8122932671
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

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

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

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

void printReverseOrder(Node* tail) {
    Node* current = tail;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->prev;
    }
    printf("\n");
}

void freeList(Node* head) {
    Node* current = head;
    while (current != NULL) {
        Node* temp = current;
        current = current->next;
        free(temp);
    }
}

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

    Node* head = NULL;
    Node* tail = NULL;
```

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

    printf("List in original order:\n");
    printOriginalOrder(head);

    printf("List in reverse order:\n");
    printReverseOrder(tail);

    freeList(head);

    return 0;
}
```

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

2. 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

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

// Define the structure for a node in the doubly linked list
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, struct Node** tail, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        *tail = newNode;
    } else {
        (*tail)->next = newNode;
        newNode->prev = *tail;
        *tail = newNode;
    }
}

// Function to display the doubly linked list
```

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

// Function to free the memory allocated for the doubly linked list
void freeList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        struct Node* temp = current;
        current = current->next;
        free(temp);
    }
}

int main() {
    int n;
    scanf("%d", &n);
    if (n == 0) {
        printf("List is empty\n");
    } else {
        struct Node* head = NULL;
        struct Node* tail = NULL;
        for (int i = 0; i < n; i++) {
            int num;
            scanf("%d", &num);
            append(&head, &tail, num);
        }
        display(head);
        freeList(head);
    }
    return 0;
}
```

240701424

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

3. 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

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

```
// Define the structure for a node in the doubly linked list
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 end of the doubly linked list (push)
void append(struct Node** head, struct Node** tail, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        *tail = newNode;
    } else {
        (*tail)->next = newNode;
        newNode->prev = *tail;
        *tail = newNode;
    }
}

// Function to remove the first node from the doubly linked list (pop bottom)
void removeFirst(struct Node** head, struct Node** tail) {
    if (*head == NULL) {
        return;
    }
    struct Node* temp = *head;
    *head = (*head)->next;
    if (*head != NULL) {
        (*head)->prev = NULL;
    } else {
        *tail = NULL;
    }
    free(temp);
```

240701424

```
}

// Function to print the doubly linked list from head to tail (bottom to top)
void printList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");
}

// Function to free the memory allocated for the doubly linked list
void freeList(struct Node* head) {
    struct Node* current = head;
    while (current != NULL) {
        struct Node* temp = current;
        current = current->next;
        free(temp);
    }
}

int main() {
    int n;
    scanf("%d", &n);
    struct Node* head = NULL;
    struct Node* tail = NULL;
    for (int i = 0; i < n; i++) {
        int ticket;
        scanf("%d", &ticket);
        append(&head, &tail, ticket);
    }
    // Remove the first inserted ticket (bottom of the stack)
    removeFirst(&head, &tail);
    // Print the remaining tickets from bottom to top
    printList(head);
    freeList(head);
    return 0;
}
```

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