Rajalakshmi Engineering College

Name: Grisha P
Email: 241901030@rajalakshmi.edu.in
Roll no: 241901030
Phone: 9150371403
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

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>

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

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

void insert(struct Node** head, int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = NULL;
   newNode->prev = NULL;
```

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

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

    struct Node* head = NULL;
    for (int i = 0; i < n; i++) {
        int value;
        scanf("%d", &value);
        insert(&head, value);
    }

    display(head);
    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>

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

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

```
    if (n == 0) {
        printf("List in original order:\n");
        printf("\n");
        printf("List in reverse order:\n");
        printf("\n");
        return 0;
    }

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

    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
        newNode->data = value;
        newNode->prev = NULL;
        newNode->next = NULL;

        if (head == NULL) {
            head = newNode;
            tail = newNode;
        } else {
            tail->next = newNode;
            newNode->prev = tail;
            tail = newNode;
        }
    }

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

    printf("List in reverse order:\n");
    temp = tail;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->prev;
    }
    printf("\n");

    return 0;
}
```

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>

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

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

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

   for (int i = 0; i < n; i++) {
      scanf("%d", &value);
      struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
      newNode->data = value;
      newNode->prev = NULL;
      newNode->next = NULL;

      if (head == NULL) {
         head = newNode;
         tail = newNode;
      } else {
         tail->next = newNode;
         newNode->prev = tail;
         tail = newNode;
      }
   }

   if (head != NULL) {
      struct Node* temp = head;
      head = head->next;
      if (head != NULL)
         head->prev = NULL;
      free(temp);
```

```
    }

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

    return 0;
}
```

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