Rajalakshmi Engineering College

Name: KEERTHI PRIYA T
Email: 240701258@rajalakshmi.edu.in
Roll no: 2116240701258
Phone: 7397397221
Branch: REC
Department: I CSE FC
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>

// Define the Node structure
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 = newNode->next = NULL;
    return newNode;
}

// Function to append node at the end
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;
    }
}

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

// Function to print list from tail to head
void printReverse(Node* tail) {
    Node* temp = tail;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->prev;
    }
}

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

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

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

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

```
    printf("\n");

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

    return 0;
}
```

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

2. 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

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

// Node structure
typedef struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
} Node;

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

// Insert at beginning
void insertAtBeginning(Node** head, int data) {
    Node* newNode = createNode(data);
    newNode->next = *head;
    if (*head != NULL)
        (*head)->prev = newNode;
    *head = newNode;
}

// Insert at end
void insertAtEnd(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;
```

```
    }
}

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

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

    int arr[n];
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    // Insert at beginning
    Node* headBegin = NULL;
    for (int i = 0; i < n; i++)
        insertAtBeginning(&headBegin, arr[i]);
    printList(headBegin);

    // Insert at end
    Node *headEnd = NULL, *tailEnd = NULL;
    for (int i = 0; i < n; i++)
        insertAtEnd(&headEnd, &tailEnd, arr[i]);
    printf(" ");
    printList(headEnd);

    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

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

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

```
    struct Node* next;
} Node;

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

// Push: add at the end (top of stack)
void push(Node** head, Node** tail, int data) {
    Node* newNode = createNode(data);
    if (*tail == NULL) {
        *head = *tail = newNode;
    } else {
        (*tail)->next = newNode;
        newNode->prev = *tail;
        *tail = newNode;
    }
}

// Remove the oldest ticket: from the bottom (head)
void revokeOldest(Node** head, Node** tail) {
    if (*head == NULL) return;

    Node* temp = *head;
    *head = (*head)->next;
    if (*head)
        (*head)->prev = NULL;
    else
        *tail = NULL;  // If list becomes empty

    free(temp);
}

// Print from bottom (head) to top (tail)
void printTickets(Node* head) {
    Node* temp = head;
    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 ticket;
        scanf("%d", &ticket);
        push(&head, &tail, ticket);
    }

    revokeOldest(&head, &tail);  // Remove first inserted ticket

    printTickets(head);
    printf("\n");

    return 0;
}
```

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

2116240701258