Rajalakshmi Engineering College

Name: Soameshwaran D
Email: 240701520@rajalakshmi.edu.in
Roll no: 240701520
Phone: 7358671540
Branch: REC
Department: I CSE FE
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 : 10

Section 1 : Coding

1. Problem Statement

Imagine you're managing a store's inventory list, and some products were accidentally entered multiple times. You need to remove the duplicate products from the list to ensure each product appears only once.

You have an unsorted doubly linked list of product IDs. Some of these product IDs may appear more than once, and your goal is to remove any duplicates.

Input Format

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

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

Output Format

The output prints the final after removing duplicate nodes, separated by a space.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 10
12 12 10 4 8 4 6 4 4 8

Output: 8 4 6 10 12

Answer

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

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

```
// Function to check if value exists in the list
int exists(Node* head, int data) {
    while (head != NULL) {
        if (head->data == data)
            return 1;
        head = head->next;
    }
    return 0;
}

// Function to free the list
void freeList(Node* head) {
    while (head != NULL) {
        Node* temp = head;
        head = head->next;
        free(temp);
    }
}

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

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

    Node* uniqueList = NULL;

    // Traverse from end to start to preserve reverse first appearance
    for (int i = n - 1; i >= 0; i--) {
        if (!exists(uniqueList, arr[i])) {
            insertAtBeginning(&uniqueList, arr[i]);
        }
    }

    // Print the list
    Node* current = uniqueList;
    while (current != NULL) {
        printf("%d ", current->data);
```

```
        current = current->next;
    }

    // Free allocated memory
    freeList(uniqueList);
    return 0;
}
```

Status : Wrong ***Marks : 0/10***

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 using GCC
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
    int ticket;
    struct Node* prev;
    struct Node* next;
} Node;

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

// Push to top (head)
void push(int ticket) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->ticket = ticket;
    newNode->prev = NULL;
    newNode->next = head;

    if (head != NULL)
        head->prev = newNode;
    else
        tail = newNode;  // First node becomes tail too

    head = newNode;
}

// Remove from bottom (tail)
void removeBottom() {
    if (tail == NULL)
        return;

    Node* temp = tail;
    if (tail->prev) {
        tail = tail->prev;
        tail->next = NULL;
```

```
    } else {
        // Only one node
        head = NULL;
        tail = NULL;
    }
    free(temp);
}

// Print from bottom to top (tail to head)
void printTickets() {
    Node* current = tail;
    while (current != NULL) {
        printf("%d ", current->ticket);
        current = current->prev;
    }
}

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

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

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

    // Remove the oldest ticket (bottom)
    removeBottom();

    // Print remaining tickets from bottom to top
    printTickets();

    // Free memory
```

```
    freeList();

    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

-

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