Rajalakshmi Engineering College

Name: Pooja S
Email: 240701386@rajalakshmi.edu.in
Roll no: 240701386
Phone: 8838480229
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 : 2
Total Mark : 30
Marks Obtained : 30

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>

#define MAX 101

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

// Insert at the front of the DLL
void push(struct Node** head_ref, int new_data) {
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
    new_node->data = new_data;
    new_node->prev = NULL;
    new_node->next = *head_ref;

    if (*head_ref != NULL)
        (*head_ref)->prev = new_node;

    *head_ref = new_node;
}

// Print the list
```

```
void printList(struct Node* node) {
    while (node != NULL) {
        printf("%d ", node->data);
        node = node->next;
    }
}

// Remove duplicates using hash table
void removeDuplicates(struct Node** head_ref) {
    int seen[MAX] = {0};  // assuming 1 ≤ data ≤ 100
    struct Node* curr = *head_ref;
    struct Node* next_node;

    while (curr != NULL) {
        if (seen[curr->data]) {
            // Duplicate found, remove it
            struct Node* temp = curr;
            if (curr->prev)
                curr->prev->next = curr->next;
            if (curr->next)
                curr->next->prev = curr->prev;
            if (curr == *head_ref)
                *head_ref = curr->next;
            curr = curr->next;
            free(temp);
        } else {
            seen[curr->data] = 1;
            curr = curr->next;
        }
    }
}

int main() {
    int n, val;
    struct Node* head = NULL;

    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &val);
        push(&head, val);  // insert at front to maintain reverse order
    }
```

```
    removeDuplicates(&head);
    printList(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>
```

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

// Append node to the end of DLL
void append(struct Node** head_ref, int value) {
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
    struct Node* last = *head_ref;

    new_node->data = value;
    new_node->next = NULL;

    if (*head_ref == NULL) {
        new_node->prev = NULL;
        *head_ref = new_node;
        return;
    }

    while (last->next != NULL)
        last = last->next;

    last->next = new_node;
    new_node->prev = last;
}

// Display the list
void displayList(struct Node* node) {
    if (node == NULL) {
        printf("List is empty\n");
        return;
    }

    while (node != NULL) {
        printf("%d ", node->data);
        node = node->next;
    }
    printf("\n");
}
```

```
int main() {
    int n, val;
    struct Node* head = NULL;

    scanf("%d", &n);

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

    displayList(head);

    return 0;
}
```

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

3. Problem Statement

Aarav is working on a program to analyze his test scores, which are stored in a doubly linked list. He needs a solution to input scores into the list and determine the highest score.

Help him by providing code that lets users enter test scores into the doubly linked list and find the maximum score efficiently.

Input Format

The first line 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 score to be inserted.

Output Format

The output prints an integer, representing the highest score present in the list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
89 71 2 70

Output: 89

Answer

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

// Define a Node
struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};

// Function to append a node at the end
void append(struct Node** head_ref, int value) {
    struct Node* new_node = (struct Node*) malloc(sizeof(struct Node));
    struct Node* last = *head_ref;

    new_node->data = value;
    new_node->next = NULL;

    if (*head_ref == NULL) {
        new_node->prev = NULL;
        *head_ref = new_node;
        return;
    }

    while (last->next != NULL)
        last = last->next;

    last->next = new_node;
    new_node->prev = last;
}

// Function to find the maximum score
```

```
int findMax(struct Node* head) {
    int max = head->data;

    while (head != NULL) {
        if (head->data > max)
            max = head->data;
        head = head->next;
    }
    return max;
}

int main() {
    int N, val;
    struct Node* head = NULL;

    scanf("%d", &N);

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

    int maxScore = findMax(head);
    printf("%d\n", maxScore);

    return 0;
}
```

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