Rajalakshmi Engineering College

Name: Jhanani shree
Email: 240701215@rajalakshmi.edu.in
Roll no: 240701215
Phone: 7373333511
Branch: REC
Department: I CSE AH
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* next;
    struct Node* prev;
} Node;
Node* createNode(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}
void insertEnd(Node** head, int data) {
    Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->prev = temp;
}
void removeDuplicates(Node** head) {
    Node* current = *head;
    Node* temp;
    int hash[101] = {0};

    while (current != NULL) {
        if (hash[current->data] == 0) {
            hash[current->data] = 1;
        } else {
            if (current->prev) {
                current->prev->next = current->next;
            }
            if (current->next) {
                current->next->prev = current->prev;
            }
            temp = current;
            current = current->next;
            free(temp);
            continue;
        }
        current = current->next;
    }
}

void printList(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
int main() {
    int n;
    scanf("%d", &n);

    Node* head = NULL;
    int value;

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

    removeDuplicates(&head);
    printList(head);
    Node* current = head;
    while (current != NULL) {
        Node* next = current->next;
        free(current);
        current = next;
    }

    return 0;
}
```

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

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

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

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

void insertAtEnd(Node** head, int data) {
   Node* newNode = createNode(data);
   if (*head == NULL) {
      *head = newNode;
```

```
        return;
    }
    Node* temp = *head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->prev = temp;
}
void printList(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// Function to print the list from tail to head
void printReverseList(Node* head) {
    if (head == NULL) return;
    Node* temp = head;
    while (temp->next != NULL) {
        temp = temp->next;
    }
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->prev;
    }
    printf("\n");
}

// Main function
int main() {
    int N;
    scanf("%d", &N);

    Node* head = NULL;
    for (int i = 0; i < N; i++) {
        int value;
        scanf("%d", &value);
        insertAtBeginning(&head, value);
    }
    printReverseList(head);

    head = NULL;

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

    // Free the allocated memory
    Node* current = head;
    while (current != NULL) {
        Node* next = current->next;
        free(current);
        current = next;
    }

    return 0;
}
```

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

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

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

// Define the structure for a doubly linked list node
typedef struct Node {
   int data;
   struct Node* next;
   struct Node* prev;
} Node;

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

// Function to insert a node at the end of the doubly linked list
void insertAtEnd(Node** head, int data) {
   Node* newNode = createNode(data);
   if (*head == NULL) {
      *head = newNode;
      return;
   }
```

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

// Function to print the list from head to tail
void printList(Node* head) {
    if (head == NULL) {
        printf("List is empty\n");
        return;
    }
    Node* temp = head;
    while (temp != NULL) {
        printf("%d", temp->data);
        if (temp->next != NULL) {
            printf(" ");
        }
        temp = temp->next;
    }
    printf("\n");
}

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

    Node* head = NULL;

    // Read the elements and insert them into the doubly linked list
    for (int i = 0; i < n; i++) {
        int value;
        scanf("%d", &value);
        insertAtEnd(&head, value);
    }

    // Print the list
    printList(head);
    // Free the allocated memory
    Node* current = head;
    while (current != NULL) {
        Node* next = current->next;
        free(current);
        current = next;
    }

    return 0;
}
```

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