Rajalakshmi Engineering College

Name: bharath kumar
Email: 241801032@rajalakshmi.edu.in
Roll no: 2116241801032
Phone: 7305320010
Branch: REC
Department: I AI & DS FA
Batch: 2028
Degree: B.E - AI & DS

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

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

```
#include <stdio.h>
#include <stdlib.h>

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

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

// Function to remove duplicates from the doubly linked list
void removeDuplicates(struct Node* head) {
    struct Node *current = head, *temp, *runner;

    while (current != NULL) {
        runner = current->next;
        while (runner != NULL) {
            if (current->data == runner->data) {
```

```
            // Duplicate found, remove runner node
            temp = runner;
            runner->prev->next = runner->next;
            if (runner->next != NULL) {
                runner->next->prev = runner->prev;
            }
            runner = runner->next;
            free(temp);
        } else {
            runner = runner->next;
        }
    }
    current = current->next;
  }
}

// Function to print the list
void printList(struct Node* head) {
  struct Node* current = head;
  while (current != NULL) {
    printf("%d ", current->data);
    current = current->next;
  }
  printf("\n");
}

// Function to insert a node at the beginning of the list
void insertAtBeginning(struct Node** head, int data) {
  struct Node* newNode = createNode(data);
  newNode->next = *head;
  if (*head != NULL) {
    (*head)->prev = newNode;
  }
  *head = newNode;
}

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

  // Read the number of elements
  scanf("%d", &n);
```

```
    // Read the elements and insert them at the beginning of the list
    for (int i = 0; i < n; i++) {
        scanf("%d", &val);
        insertAtBeginning(&head, val);
    }

    // Remove duplicates
    removeDuplicates(head);

    // Print the list after removing duplicates
    printList(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

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

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

// Function to create a new node
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct 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(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
```

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

// Function to print the list in original order
void printOriginalOrder(struct Node* head) {
    struct Node* temp = head;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

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

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

    // Read the number of elements
    scanf("%d", &n);

    // Read the elements and insert them at the end of the list
    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
```

```
        insertAtEnd(&head, data);
    }

    // Print the list in original order
    printf("List in original order:\n");
    printOriginalOrder(head);

    // Print the list in reverse order
    printf("List in reverse order:\n");
    printReverseOrder(head);

    return 0;
}
```

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

3. 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

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
    struct Node* prev;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    newNode->prev = NULL;
    return newNode;
}
struct Node* insertAtBeginning(struct Node* head, int data) {
    struct Node* newNode = createNode(data);
    if (head == NULL) {
        return newNode;
    }
    newNode->next = head;
    head->prev = newNode;
    return newNode;
}
struct Node* insertAtEnd(struct Node* head, int data) {
    struct Node* newNode = createNode(data);
    if (head == NULL) {
        return newNode;
    }
    struct Node* temp = head;
    while (temp->next != NULL) {
```

```
        temp = temp->next;
    }
    temp->next = newNode;
    newNode->prev = temp;
    return head;
}

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

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

    int* values = (int*)malloc(N * sizeof(int));
    for (int i = 0; i < N; i++) {
        scanf("%d", &values[i]);
    }

    struct Node* head = NULL;
    for (int i = 0; i <N; i++) {
        head = insertAtBeginning(head, values[i]);
    }
    printList(head);
    printf("\n");
    struct Node* headEnd = NULL;
    for (int i = 0; i <N; i++) {
        headEnd = insertAtEnd(headEnd, values[i]);
    }
    printList(headEnd);
    printf("\n");
    free(values);

    return 0;
}
```

Status : Correct *Marks : 10/10*