Rajalakshmi Engineering College

Name: Padma Priya D
Email: 240701377@rajalakshmi.edu.in
Roll no: 240701377
Phone: 8668123104
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 : 1
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. 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>
struct Node {
   int data;
   struct Node* prev;
   struct Node* next;
};
void insertAtBeginning(struct Node** head_ref, int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->prev = NULL;
   newNode->next = *head_ref;

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

   *head_ref = newNode;
}
void insertAtEnd(struct Node** head_ref, int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = NULL;
```

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

    struct Node* temp = *head_ref;
    while (temp->next != NULL)
        temp = temp->next;

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

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

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

    struct Node* headBegin = NULL;
    struct Node* headEnd = NULL;
    for (int i = 0; i < N; i++)
        insertAtBeginning(&headBegin, arr[i]);
    for (int i = 0; i < N; i++)
        insertAtEnd(&headEnd, arr[i]);
    printList(headBegin);
    printList(headEnd);
return 0;
}
```

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

2. 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>
#include <stdbool.h>
struct Node {
   int data;
   struct Node* prev;
   struct Node* next;
};
```

```
void insertAtBeginning(struct Node** head_ref, int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->prev = NULL;
   newNode->next = *head_ref;

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

   *head_ref = newNode;
}
void removeDuplicates(struct Node** head_ref) {
   bool seen[101] = {false};
   struct Node* curr = *head_ref;
   while (curr != NULL) {
      if (seen[curr->data]) {
         struct Node* toDelete = curr;
         struct Node* nextNode = curr->next;

         if (toDelete->prev)
            toDelete->prev->next = toDelete->next;
         else
            *head_ref = toDelete->next;

         if (toDelete->next)
            toDelete->next->prev = toDelete->prev;

         free(toDelete);
         curr = nextNode;
      } else {
         seen[curr->data] = true;
         curr = curr->next;
      }
   }
}
void printList(struct Node* head) {
   struct Node* temp = head;
   while (temp != NULL) {
      printf("%d ", temp->data);
      temp = temp->next;
   }
   printf("\n");
}
int main() {
   int n;
   scanf("%d", &n);

   struct Node* head = NULL;

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

   removeDuplicates(&head);
   printList(head);

   return 0;
}
```

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

3. Problem Statement

Imagine Anu is tasked with finding the middle element of a doubly linked list. Given a doubly linked list where each node contains an integer value and is inserted at the end, implement a program to find the middle element of the list. If the number of nodes is even, return the middle element pair.

Input Format

The first line of input consists of an integer N, representing the number of nodes in the doubly linked list.

The second line consists of N space-separated integers, representing the values of the nodes in the doubly linked list.

Output Format

The first line of output prints the space-separated elements of the doubly linked list.

The second line prints the middle element(s) of the doubly linked list, depending on whether the number of nodes is odd or even.

Refer to the sample outputs for the formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50

Output: 10 20 30 40 50
30

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
struct Node {
   int data;
   struct Node* prev;
   struct Node* next;
};
void insertAtEnd(struct Node** head_ref, int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = NULL;

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

   struct Node* temp = *head_ref;
   while (temp->next != NULL)
      temp = temp->next;

   temp->next = newNode;
   newNode->prev = temp;
}
void printList(struct Node* head) {
   struct Node* temp = head;
```

```
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
void printMiddle(struct Node* head, int N) {
    struct Node* temp = head;
    int count = 0;
    int mid1 = N / 2;
    int mid2 = (N % 2 == 0) ? mid1 - 1 : mid1;

    for (int i = 0; i <= mid1; i++) {
        if (i == mid2)
            printf("%d", temp->data);
        if (N % 2 == 0 && i == mid1)
            printf(" %d", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

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

    struct Node* head = NULL;

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

    return 0;
}
```

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