# Rajalakshmi Engineering College

Name: Naveed Sheriff
Email: 240701348@rajalakshmi.edu.in
Roll no: 240701348
Phone: 9025573780
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

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>

struct node
{
   int data;
   struct node*prev;
   struct node*next;
};

struct node*head = NULL;

void insert(int e)
{
   struct node*newnode = (struct node*)malloc(sizeof(struct node));
   newnode->data = e;
   newnode->prev = NULL;
   newnode->next = NULL;

   if(head == NULL)
   head = newnode;
```

```
    else{
        struct node*temp = head;
        while(temp->next != NULL)
        {
            temp = temp->next;
        }
        newnode->prev = temp;
        temp->next = newnode;
    }
}
void display()
{
    struct node*temp = head->next;
    while(temp!= NULL)
    {
        printf("%d ",temp->data);
        temp = temp->next;
    }
}
int main()
{
    int n,e;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        scanf("%d",&e);
        insert(e);
    }
    display();
}
```

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

2. 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;
};

struct node*head = NULL;

void insertatend(int e)
{
   struct node*newnode = (struct node*)malloc(sizeof(struct node));
   newnode->data = e;
   newnode->prev = NULL;
   newnode->next = NULL;

   if(head == NULL)
   head = newnode;

   else{
       struct node*temp = head;
       while(temp->next != NULL)
       {
           temp = temp->next;
       }
       newnode->prev = temp;
       temp->next = newnode;
   }
}

void findmid(int n)
{
   int i=0;
   struct node*temp = head;
   if(n%2 == 0)
   {
       while(i <= n/2)
       {
           if(i == n/2)
           printf("%d ",temp->data);

           else if(i == (n-1)/2)
           printf("%d ",temp->data);

           temp = temp->next;
           i += 1;
       }
   }
   else{
       while(i <= n/2)
       {
           if(i == n/2)
           printf("%d",temp->data);
           temp=temp->next;
           i += 1;
       }
   }
}
void display()
{
   struct node*temp = head;
   while(temp!= NULL)
   {
       printf("%d ",temp->data);
       temp = temp->next;
   }
   printf("\n");
}
int main()
{
   int n,e;
   scanf("%d",&n);
   for(int i=0 ; i<n ;i++)
   {
       scanf("%d",&e);
       insertatend(e);
   }
   display();
   findmid(n);
}
```

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

3. 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>

struct node
{
   int data;
   struct node*prev;
   struct node*next;
};

struct node*head = NULL;

void insertatend(int e)
{
   struct node*newnode = (struct node*)malloc(sizeof(struct node));
   newnode->data = e;
   newnode->prev = NULL;
   newnode->next = NULL;
```

```
    if(head == NULL)
    head = newnode;

    else{
        struct node*temp = head;
        while(temp->next != NULL)
            temp = temp->next;
        temp->next = newnode;
        newnode->prev = temp;
    }
}

void finddup()
{
    int seen[101] = {0};
    struct node*curr = head;
    struct node*temp ;
    while (curr && curr->next)
        curr = curr->next;
    while(curr)
    {
        if(seen[curr->data])
        {
            struct node*todelete = curr;
            if(curr->prev)
                curr->prev->next = curr->next;
            if(curr->next)
                curr->next->prev = curr->prev;
            if(curr == head)
                head = curr->next;
            temp = curr->prev;
            free(todelete);
            curr = temp;
        }
        else
        {
            seen[curr->data] = 1;
            curr = curr->prev;
        }
    }
}
void display()
```

```
{
   struct node*temp = head;
   while(temp->next!= NULL)
   {
      temp = temp->next;
   }
   while(temp!= NULL)
   {
      printf("%d ",temp->data);
      temp = temp->prev;
   }
}
int main()
{
   int n,e;
   scanf("%d",&n);
   for(int i=1;i<=n; i++)
   {
      scanf("%d",&e);
      insertatend(e);
   }
   finddup();
   display();

}
```

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