# Rajalakshmi Engineering College

Name: Prajeet V
Email: 240701389@rajalakshmi.edu.in
Roll no: 240701389
Phone: 9363389322
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 4_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 30

## Section 1 : Coding

1. Problem Statement

A customer support system is designed to handle incoming requests using a queue. Implement a linked list-based queue where each request is represented by an integer. After processing the requests, remove any duplicate requests to ensure that each request is unique and print the remaining requests.

***Input Format***

The first line of input consists of an integer N, representing the number of requests to be enqueued.

The second line consists of N space-separated integers, each representing a request.

***Output Format***

The output prints space-separated integers after removing the duplicate requests.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 5
2 4 2 7 5

Output: 2 4 7 5

***Answer***

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

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

Node* front = NULL;
Node* rear = NULL;

// Function to enqueue (add) a request
void enqueue(int data) {
    Node* new_node = (Node*)malloc(sizeof(Node));
    new_node->data = data;
    new_node->next = NULL;

    if (!rear) {  // If queue is empty
        front = new_node;
        rear = new_node;
    } else {
        rear->next = new_node;
        rear = new_node;
    }
}
```

```
// Function to remove duplicates
void removeDuplicates() {
    Node* current = front;
    while (current) {
        Node* prev = current;
        Node* temp = current->next;
        while (temp) {
            if (temp->data == current->data) {
                prev->next = temp->next;
                free(temp);
                temp = prev->next;
            } else {
                prev = temp;
                temp = temp->next;
            }
        }
        current = current->next;
    }
}

// Function to display the queue
void displayQueue() {
    Node* temp = front;
    while (temp) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

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

    for (int i = 0; i < n; i++) {
        scanf("%d", &request);
        enqueue(request);
    }

    // Remove duplicates
    removeDuplicates();
```

```
    // Display final queue
    displayQueue();

    return 0;
}
```

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

2. Problem Statement

Fathima has been tasked with developing a program to manage a queue of customers waiting in line at a service center. Help her write a program simulating a queue data structure using a linked list.

Here is a description of the scenario and the required operations:

Enqueue: Add a customer to the end of the queue.Dequeue: Remove and discard a customer from the front of the queue.Display waiting customers: Display the front and rear customer IDs in the queue.

Write a program that enqueues all the customers into the queue, performs a dequeue operation, and prints the front and rear elements.

***Input Format***

The first input line consists of an integer N, representing the number of customers to be inserted into the queue.

The second line consists of N space-separated integers, representing the customer IDs.

***Output Format***

The output prints "Front: X, Rear: Y" where X is the front element and Y is the rear element, after performing the dequeue operation.

Refer to the sample output for the exact text and format.

***Sample Test Case***

Input: 5
112 104 107 116 109
Output: Front: 104, Rear: 109

***Answer***

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

struct Node {
    int customerID;
    struct Node* next;
};

struct Queue {
    struct Node* front;
    struct Node* rear;
};

void initQueue(struct Queue* queue) {
    queue->front = NULL;
    queue->rear = NULL;
}

void enqueue(struct Queue* queue, int customerID) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->customerID = customerID;
    newNode->next = NULL;

    if (queue->rear == NULL) {
        queue->front = newNode;
        queue->rear = newNode;
    } else {
        queue->rear->next = newNode;
        queue->rear = newNode;
    }
}

void dequeue(struct Queue* queue) {
    if (queue->front == NULL) {
        printf("Queue is empty!\n");
        return;
```

```
    }

    struct Node* temp = queue->front;
    queue->front = queue->front->next;

    if (queue->front == NULL) {
        queue->rear = NULL;
    }

    free(temp);
}

void displayQueue(struct Queue* queue) {
    if (queue->front == NULL) {
        printf("Queue is empty!\n");
        return;
    }

    printf("Front: %d, Rear: %d\n", queue->front->customerID, queue->rear->customerID);
}

int main() {
    struct Queue queue;
    initQueue(&queue);

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

    int customerID;
    for (int i = 0; i < N; i++) {
        scanf("%d", &customerID);
        enqueue(&queue, customerID);
    }

    dequeue(&queue);

    displayQueue(&queue);

    return 0;
}
```

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

3. Problem Statement

Sara builds a linked list-based queue and wants to dequeue and display all positive even numbers in the queue. The numbers are added at the end of the queue.

Help her by writing a program for the same.

***Input Format***

The first line of input consists of an integer N, representing the number of elements Sara wants to add to the queue.

The second line consists of N space-separated integers, each representing an element to be enqueued.

***Output Format***

The output prints space-separated the positive even integers from the queue, maintaining the order in which they were enqueued.

Refer to the sample output for formatting specifications.

***Sample Test Case***

Input: 5
1 2 3 4 5
Output: 2 4

***Answer***

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

struct Node {
    int data;
    struct Node* next;
};
```

```
struct Queue {
    struct Node* front;
    struct Node* rear;
};

void initQueue(struct Queue* queue) {
    queue->front = NULL;
    queue->rear = NULL;
}

void enqueue(struct Queue* queue, int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->next = NULL;

    if (queue->rear == NULL) {
        queue->front = newNode;
        queue->rear = newNode;
    } else {
        queue->rear->next = newNode;
        queue->rear = newNode;
    }
}

void dequeueAndDisplayEven(struct Queue* queue) {
    if (queue->front == NULL) {
        printf("\nQueue is empty!\n");
        return;
    }

    struct Node* temp = queue->front;
    int firstEven = 1;

    while (temp != NULL) {
        if (temp->data > 0 && temp->data % 2 == 0) {
            if (firstEven) {
                printf("%d", temp->data);
                firstEven = 0;
            } else {
                printf(" %d", temp->data);
            }
```

```
        }
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    struct Queue queue;
    initQueue(&queue);

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

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

    dequeueAndDisplayEven(&queue);

    return 0;
}
```

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