Rajalakshmi Engineering College

Name: SUBASH R
Email: 240701538@rajalakshmi.edu.in
Roll no: 240701538
Phone: 9150202710
Branch: REC
Department: I CSE FE
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

Pathirana is a medical lab specialist who is responsible for managing blood count data for a group of patients. The lab uses a queue-based system to track the blood cell count of each patient. The queue structure helps in processing the data in a first-in-first-out (FIFO) manner.

However, Pathirana needs to remove the blood cell count that is positive even numbers from the queue using array implementation of queue, as they are not relevant to the specific analysis he is performing. The remaining data will then be used for further medical evaluations and reporting.

Input Format

The first line consists of an integer n, representing the number of a patient's

blood cell count.

The second line consists of n space-separated integers, representing a blood cell count value.

Output Format

The output displays space-separated integers, representing the remaining blood cell count after removing the positive even numbers.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
1 2 3 4 5

Output: 1 3 5

Answer

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

#define MAX_SIZE 100

typedef struct {
    int data[MAX_SIZE];
    int front;
    int rear;
} Queue;

void initializeQueue(Queue *q) {
    q->front = 0;
    q->rear = -1;
}

int isEmpty(Queue *q) {
    return (q->front > q->rear);
}

int isFull(Queue *q) {
```

```
    return (q->rear == MAX_SIZE - 1);
}

void enqueue(Queue *q, int value) {
    if (isFull(q)) {
        printf("Queue is full. Cannot enqueue.\n");
        return;
    }
    q->rear++;
    q->data[q->rear] = value;
}

int dequeue(Queue *q) {
    if (isEmpty(q)) {
        printf("Queue is empty. Cannot dequeue.\n");
        return -1;
    }
    int value = q->data[q->front];
    q->front++;
    return value;
}

void removeEvenPositive(Queue *q) {
    Queue tempQueue;
    initializeQueue(&tempQueue);
    while (!isEmpty(q)) {
        int value = dequeue(q);
        if (value <= 0 || value % 2 != 0) {
            enqueue(&tempQueue, value);
        }
    }
    q->front = 0;
    q->rear = -1;
    while (!isEmpty(&tempQueue)) {
        enqueue(q, dequeue(&tempQueue));
    }
}

void displayQueue(Queue *q) {
    if (isEmpty(q)) {
        printf("Queue is empty.\n");
        return;
```

```
    }
    for (int i = q->front; i <= q->rear; i++) {
        printf("%d ", q->data[i]);
    }
    printf("\n");
}

int main() {
    Queue bloodCounts;
    initializeQueue(&bloodCounts);
    int n, count;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &count);
        enqueue(&bloodCounts, count);
    }
    removeEvenPositive(&bloodCounts);
    displayQueue(&bloodCounts);
    return 0;
}
```

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

2. Problem Statement

Saran is developing a simulation for a theme park where people wait in a queue for a popular ride.

Each person has a unique ticket number, and he needs to manage the queue using a linked list implementation.

Your task is to write a program for Saran that reads the number of people in the queue and their respective ticket numbers, enqueue them, and then calculate the sum of all ticket numbers to determine the total ticket value present in the queue.

Input Format

The first line of input consists of an integer N, representing the number of people in the queue.

The second line consists of N space-separated integers, representing the ticket numbers.

Output Format

The output prints an integer representing the sum of all ticket numbers.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
2 4 6 7 5
Output: 24

Answer

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

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

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

struct Queue* createQueue() {
    struct Queue* q = (struct Queue*)malloc(sizeof(struct Queue));
    q->front = q->rear = NULL;
    return q;
}

void enqueue(struct Queue* q, int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    if (q->rear == NULL) {
        q->front = q->rear = newNode;
```

```
        return;
    }
    q->rear->next = newNode;
    q->rear = newNode;
}

int main() {
    int n, ticket;
    struct Queue* queue = createQueue();
    long long sum = 0;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &ticket);
        enqueue(queue, ticket);
        sum += ticket;
    }
    printf("%lld\n", sum);
    struct Node* current = queue->front;
    while (current != NULL) {
        struct Node* temp = current;
        current = current->next;
        free(temp);
    }
    free(queue);
    return 0;
}
```

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

3. 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

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

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

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

void enqueue(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (newNode == NULL) {
        printf("Memory allocation failed!\n");
        exit(1);
```

```
    }
    newNode->data = data;
    newNode->next = NULL;
    if (rear == NULL) {
        front = rear = newNode;
        return;
    }
    rear->next = newNode;
    rear = newNode;
}

void dequeue() {
    if (front == NULL) {
        return;
    }
    Node* temp = front;
    front = front->next;
    if (front == NULL) {
        rear = NULL;
    }
    free(temp);
}

void display() {
    if (front == NULL) {
        printf("Queue is empty\n");
        return;
    }
    printf("Front: %d, ", front->data);
    printf("Rear: %d\n", rear->data);
}

int main() {
    int n, customer_id;
    scanf("%d", &n);
    for(int i=0; i<n; i++){
        scanf("%d", &customer_id);
        enqueue(customer_id);
    }
    dequeue();
    if(front == NULL){
        printf("Queue is empty\n");
```

```
        return 0;
    }
    display();
    return 0;
}
```

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