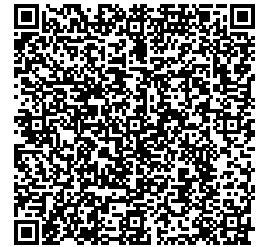# Rajalakshmi Engineering College

Name: Siddesh Kumar L
Email: 240701512@rajalakshmi.edu.in
Roll no:
Phone: null
Branch: REC
Department: I CSE FE
Batch: 2028
Degree: B.E - CSE

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

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

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

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

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

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

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

    if (q->rear == NULL) {
        q->front = q->rear = newNode;
        return;
    }

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

```c
        q->rear = newNode;
}

int dequeue(Queue* q) {
    if (q->front == NULL) return -1;

    Node* temp = q->front;
    int val = temp->data;
    q->front = q->front->next;

    if (q->front == NULL)
        q->rear = NULL;

    free(temp);
    return val;
}

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

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

    Queue q;
    initQueue(&q);

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

    while (!isEmpty(&q)) {
        int val = dequeue(&q);
        if (val > 0 && val % 2 == 0) {
            printf("%d ", val);
        }
    }

    return 0;
```

```
}
```

Status : Correct   *Marks : 10/10*

2.   Problem Statement

Imagine you are developing a basic task management system for a small
team of software developers. Each task is represented by an integer, where
positive integers indicate valid tasks and negative integers indicate
erroneous tasks that need to be removed from the queue before
processing.

Write a program using the queue with a linked list that allows the team to
add tasks to the queue, remove all erroneous tasks (negative integers), and
then display the valid tasks that remain in the queue.

Input Format

The first line consists of an integer N, representing the number of tasks to be
added to the queue.

The second line consists of N space-separated integers, representing the tasks.
Tasks can be both positive (valid) and negative (erroneous).

Output Format

The output displays the following format:

For each task enqueued, print a message "Enqueued: " followed by the task
value.

The last line displays the "Queue Elements after Dequeue: " followed by
removing all erroneous (negative) tasks and printing the valid tasks remaining in
the queue in the order they were enqueued.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
12 -54 68 -79 53

Output: Enqueued: 12
Enqueued: -54
Enqueued: 68
Enqueued: -79
Enqueued: 53
Queue Elements after Dequeue: 12 68 53

Answer

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

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

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

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

void enqueue(Queue* q, int value) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) return;

    newNode->data = value;
    newNode->next = NULL;

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

    printf("Enqueued: %d\n", value);
```

```c
}

void removeNegatives(Queue* q) {
    Node* current = q->front;
    Node* prev = NULL;

    while (current != NULL) {
        if (current->data < 0) {
            Node* toDelete = current;
            if (prev == NULL) {
                q->front = current->next;
            } else {
                prev->next = current->next;
            }

            if (current == q->rear) {
                q->rear = prev;
            }

            current = current->next;
            free(toDelete);
        } else {
            prev = current;
            current = current->next;
        }
    }
}

void displayQueue(Queue* q) {
    Node* temp = q->front;
    printf("Queue Elements after Dequeue: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

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

```
    Queue q;
    initQueue(&q);

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

    removeNegatives(&q);
    displayQueue(&q);

    return 0;
}
```

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

3. 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

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

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

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

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

void enqueue(Queue* q, int value) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) return;

    newNode->data = value;
    newNode->next = NULL;

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

```c
int sumQueue(Queue* q) {
    int sum = 0;
    Node* temp = q->front;
    while (temp != NULL) {
        sum += temp->data;
        temp = temp->next;
    }
    return sum;
}

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

    Queue q;
    initQueue(&q);

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

    int total = sumQueue(&q);
    printf("%d\n", total);

    return 0;
}
```

Status : Correct   *Marks : 10/10*