Rajalakshmi Engineering College

Name: Prithika S
Email: 240701400@rajalakshmi.edu.in
Roll no: 240701400
Phone: 9790212894
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 : 2
Total Mark : 30
Marks Obtained : 30

Section 1 : Coding

1. 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

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

// Define node structure for the queue
typedef struct Node {
    int data;
    struct Node* next;
} Node;

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

```c
// Enqueue operation
void enqueue(int value) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    if (!newNode) return;

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

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

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

// Function to remove negative (erroneous) tasks
void removeNegatives() {
    Node *current = front, *prev = NULL;

    while (current != NULL) {
        if (current->data < 0) {
            Node* temp = current;
            if (current == front) {
                front = front->next;
                if (front == NULL) rear = NULL;
                current = front;
            } else {
                prev->next = current->next;
                if (current == rear) rear = prev;
                current = current->next;
            }
            free(temp);
        } else {
            prev = current;
            current = current->next;
        }
    }
}
```

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

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

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

    removeNegatives();
    displayQueue();

    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.

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

// Node structure
typedef struct Node {
    int ticket;
    struct Node* next;
} Node;

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

// Enqueue function
void enqueue(int ticket) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->ticket = ticket;
    newNode->next = NULL;
```

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

// Function to sum all tickets
int sumTickets() {
    int sum = 0;
    Node* temp = front;
    while (temp != NULL) {
        sum += temp->ticket;
        temp = temp->next;
    }
    return sum;
}

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

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

    printf("%d\n", sumTickets());

    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

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

// Node structure for queue
typedef struct Node {
    int data;
    struct Node* next;
} Node;
```

```c
Node* front = NULL;
Node* rear = NULL;

// Enqueue function
void enqueue(int value) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = value;
    newNode->next = NULL;

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

// Dequeue function
void dequeue() {
    if (front == NULL) return; // Queue empty
    Node* temp = front;
    front = front->next;
    if (front == NULL) rear = NULL; // Queue became empty
    free(temp);
}

// Function to print front and rear elements
void printFrontRear() {
    if (front == NULL) {
        printf("Queue is empty\n");
    } else {
        printf("Front: %d, Rear: %d\n", front->data, rear->data);
    }
}

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

```
    // Perform one dequeue operation
    dequeue();

    // Print front and rear customer IDs
    printFrontRear();

    return 0;
}
```

Status : Correct   *Marks : 10/10*