Rajalakshmi Engineering College

Name: Vikashini M
Email: 241801314@rajalakshmi.edu.in
Roll no: 241801314
Phone: 9345747519
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

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

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>


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


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


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

    if (rear == NULL) {
        front = rear = newNode;
```

```
    } else {
        rear->next = newNode;
        rear = newNode;
    }
}

void dequeue() {
    if (front == NULL) {
        return;
    }

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

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

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

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

    dequeue();
    displayFrontAndRear();

    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

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Node* front = NULL;
struct Node* rear = NULL;

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

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

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

void removeErroneousTasks() {
    struct Node* current = front;
    struct Node* prev = NULL;

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

```
            }
            if (current == rear) {
                rear = prev;
            }
            struct Node* temp = current;
            current = current->next;
            free(temp);
        } else {
            prev = current;
            current = current->next;
        }
    }
}
void displayQueue() {
    struct Node* temp = front;
    printf("Queue Elements after Dequeue: ");

    if (temp == NULL) {
        printf("\n");
        return;
    }

    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
int main() {
    int N;
    scanf("%d", &N);

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

    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

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

```
// Structure for a node in the linked list
struct Node {
    int data;
    struct Node* next;
};

// Function to add a new node to the end of the queue
void enqueue(struct Node** head, struct Node** tail, int data) {
    // Create a new node
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;

    // If the queue is empty, make the new node the head and tail
    if (*head == NULL) {
        *head = newNode;
        *tail = newNode;
    } else {
        // Otherwise, add the new node to the end and update the tail
        (*tail)->next = newNode;
        *tail = newNode;
    }
}

// Function to calculate the sum of all ticket numbers in the queue
int calculateSum(struct Node* head) {
    // Initialize the sum
    int sum = 0;

    // Traverse the linked list and add the data of each node to the sum
    struct Node* current = head;
    while (current != NULL) {
        sum += current->data;
        current = current->next;
    }

    // Return the sum
    return sum;
}

int main() {
```

```
    // Get the number of people in the queue
    int N;
    scanf("%d", &N);

    // Create the head and tail pointers for the linked list
    struct Node* head = NULL;
    struct Node* tail = NULL;

    // Enqueue the ticket numbers
    for (int i = 0; i < N; i++) {
        int ticketNumber;
        scanf("%d", &ticketNumber);
        enqueue(&head, &tail, ticketNumber);
    }

    // Calculate the sum of the ticket numbers
    int sum = calculateSum(head);

    // Print the sum
    printf("%d\n", sum);

    // Free the memory allocated for the linked list
    struct Node* current = head;
    while (current != NULL) {
        struct Node* temp = current;
        current = current->next;
        free(temp);
    }

    return 0;
}
```

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