Rajalakshmi Engineering College

Name: krithika narasimhan
Email: 240701277@rajalakshmi.edu.in
Roll no: 240701277
Phone: 9677451731
Branch: REC
Department: I CSE FC
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

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 d) {
    struct Node* new_n = (struct Node*)malloc(sizeof(struct Node));
    new_n->data = d;
    new_n->next = NULL;
    if (front == NULL && rear == NULL) {
        front = rear = new_n;
    } else {
        rear->next = new_n;
        rear = new_n;
    }
```

```
}

void printFrontRear() {
    printf("Front: %d, ", front->data);
    printf("Rear: %d\n", rear->data);

}

void dequeue() {
    struct Node* temp;
    temp = front;
    front = front->next;
    free(temp);
}

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

    dequeue();
    printFrontRear();

    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>
```

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

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

Queue q;  // Global queue

void initQueue() {
    q.front = q.rear = NULL;
}

int isEmpty() {
    return q.front == NULL;
}

void enqueue(int data) {
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode->data = data;
    newNode->next = NULL;
    if (q.rear == NULL) {
        q.front = q.rear = newNode;
    } else {
        q.rear->next = newNode;
        q.rear = newNode;
    }
    printf("Enqueued: %d\n", data);
}

void dequeueAllNegative() {
    if (isEmpty()) {
        return;
    }

    while (q.front != NULL && q.front->data < 0) {
        Node* temp = q.front;
        q.front = q.front->next;
        free(temp);
```

```
    }

    if (q.front == NULL) {
        q.rear = NULL;
        return;
    }

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

void display() {
    Node* temp = q.front;
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
}

int main() {
    int n, value;
    initQueue();

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

    dequeueAllNegative();  //  Called without arguments
```

```
    printf("Queue Elements after Dequeue: ");
    display();

    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

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

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

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

struct Queue* queue = NULL;

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

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

void enqueue(int data) {
    struct Node* newNode = createNode(data);
    if (queue->rear == NULL) {
        queue->front = queue->rear = newNode;
    } else {
        queue->rear->next = newNode;
        queue->rear = newNode;
    }
}

int dequeue() {
    if (queue->front == NULL) {
        return -1;
    }
    int data = queue->front->data;
```

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

void dequeueEvens() {
    while (queue->front != NULL) {
        int num = dequeue();
        if (num % 2 == 0 && num > 0) {
            printf("%d ", num);
        }
    }
}

int main() {
    int capacity;
    scanf("%d", &capacity);
    queue = createQueue();
    for (int i = 0; i < capacity; ++i) {
        int num;
        scanf("%d", &num);
        enqueue(num);
    }
    dequeueEvens();
    free(queue);
    return 0;
}
```

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