Rajalakshmi Engineering College

Name: Thavaneshwaran s
Email: 241501231@rajalakshmi.edu.in
Roll no: 241501231
Phone: 7824883366
Branch: REC
Department: I AI & ML FC
Batch: 2028
Degree: B.E - AI & ML

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

You are tasked with implementing basic operations on a queue data structure using a linked list.

You need to write a program that performs the following operations on a queue:

Enqueue Operation: Implement a function that inserts an integer element at the rear end of the queue.Print Front and Rear: Implement a function that prints the front and rear elements of the queue. Dequeue Operation: Implement a function that removes the front element from the queue.

Input Format

The first line of input consists of an integer N, representing the number of elements to be inserted into the queue.

The second line consists of N space-separated integers, representing the queue elements.

Output Format

The first line prints "Front: X, Rear: Y" where X is the front and Y is the rear elements of the queue.

The second line prints the message indicating that the dequeue operation (front element removed) is performed: "Performing Dequeue Operation:".

The last line prints "Front: M, Rear: N" where M is the front and N is the rear elements after the dequeue operation.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 5
12 56 87 23 45

Output: Front: 12, Rear: 45
Performing Dequeue Operation:
Front: 56, Rear: 45

Answer

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

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

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

// Function to insert an element at the rear end of the queue (enqueue operation)
void enqueue(int d) {
   // Create a new node with the given data
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
```

```
    newNode->data = d;
    newNode->next = NULL;

    if (rear == NULL) {
        // If the queue is empty, both front and rear will point to the new node
        front = rear = newNode;
    } else {
        // Otherwise, add the new node to the rear of the queue
        rear->next = newNode;
        rear = newNode;
    }
}

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

// Function to remove the front element of the queue (dequeue operation)
void dequeue() {
    if (front == NULL) {
        // If the queue is empty, print a message
        printf("Queue is empty.\n");
    } else {
        // Remove the front element and update the front pointer
        struct Node* temp = front;
        front = front->next;

        if (front == NULL) {
            // If the queue becomes empty, set rear to NULL
            rear = NULL;
        }

        // Free the memory for the removed node
        free(temp);
        // Optionally, print the dequeued element (if needed)
        // printf("Dequeued: %d\n", temp->data);
    }
}

int main() {
    int n, data;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &data);
        enqueue(data);
    }
    printFrontRear();
    printf("Performing Dequeue Operation:\n");
    dequeue();
    printFrontRear();
    return 0;
}
```

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