Rajalakshmi Engineering College

Name: Mohamed Faheem A
Email: 240801198@rajalakshmi.edu.in
Roll no: 240801198
Phone: 9952218147
Branch: REC
Department: I ECE FB
Batch: 2028
Degree: B.E - ECE

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;
```

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

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

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

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

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

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

void dequeue(struct Queue* q) {
    if (q->front == NULL) {
        printf("Queue is empty.\n");
        return;
    }

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

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

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

    struct Queue q;
    initQueue(&q);

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

    printFrontAndRear(&q);
    printf("Performing Dequeue Operation:\n");
    dequeue(&q);
    printFrontAndRear(&q);

    return 0;
}
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*