Rajalakshmi Engineering College

Name: Jhanani shree
Email: 240701215@rajalakshmi.edu.in
Roll no: 240701215
Phone: 7373333511
Branch: REC
Department: I CSE AH
Batch: 2028
Degree: B.E - CSE

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 : 0

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;
};

// Function to create a new node
struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

// Function to initialize the queue
void initializeQueue(struct Queue* q) {
    q->front = q->rear = NULL;
}

// Function to check if the queue is empty
int isEmpty(struct Queue* q) {
    return (q->front == NULL);
}

// Function to enqueue an element
void enqueue(struct Queue* q, int data) {
    struct Node* newNode = createNode(data);
    if (q->rear == NULL) {
        q->front = q->rear = newNode;
        return;
    }
    q->rear->next = newNode;
    q->rear = newNode;
}

// Function to dequeue an element
void dequeue(struct Queue* q) {
    if (isEmpty(q)) {
        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);
}

// Function to print the front and rear elements
void printFrontRear(struct Queue* q) {
    if (isEmpty(q)) {
        printf("Queue is empty.\n");
        return;
    }
    printf("Front: %d, Rear: %d\n", q->front->data, q->rear->data);
}

int main() {
    struct Queue q;
    initializeQueue(&q);

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

    printFrontRear(&q);
    printf("Performing Dequeue Operation:\n");
    dequeue(&q);
    printFrontRear(&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 : Wrong ***Marks : 0/10***