

# Rajalakshmi Engineering College

Name: suriya SM

Email: 241801284@rajalakshmi.edu.in

Roll no: 241801284

Phone: 8110855156

Branch: REC

Department: I AI & DS FD

Batch: 2028

Degree: B.E - AI & DS

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;
```

```
// You are using GCC
```

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

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

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

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

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

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

```
void enqueue(struct Queue* q, int data) {  
    struct Node* temp = newNode(data);  
    if (q->rear == NULL) {  
        q->front = q->rear = temp;  
        return;  
    }  
    q->rear->next = temp;  
    q->rear = temp;  
}
```

```
void dequeue(struct Queue* q) {  
    if (q->front == NULL)  
        return;
```

```
    struct Node* temp = q->front;  
    q->front = q->front->next;
```

```
    if (q->front == NULL)  
        q->rear = NULL;
```

```
    free(temp);  
}
```

```
void printFrontRear(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, i, val;  
    scanf("%d", &N);
```

```
    struct Queue* q = createQueue();
```

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

```
    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 :** Correct

**Marks : 10/10**