Problem Solving with Data Structures

School of Computer Science & Engg

KLE Technological University

Hubballi

Recap

- Stack Data structure
- Operations Stack
  - push, pop, isfull, isempty and display
- Implementation of stack
  - Array, Structures and linked list
- Applications of Stack
  - Infix, prefix and Postfix expression
- Recursion

Course Outcome

i. Realize abstraction of data structure and its role in the problem-solving process.

ii. Apply design thinking methodology to solve real world problems using appropriate data structure.

iii. Demonstrate programming skills through online coding platform.

iv. Work collaboratively to share knowledge, skills and experiences.

KLE TECH

KLE Technological University

Creating Value
Leveraging Knowledge

Chapter 4

Queues

Content

- Queue
  - Definition, Operations
- Different Types of queues
  - Linear Queue
  - Circular Queue
  - Priority Queue
  - Double Ended Queue
- Applications
  - Scheduling Algorithms in OS
  - E-Commerce Platforms
  - Stock Market

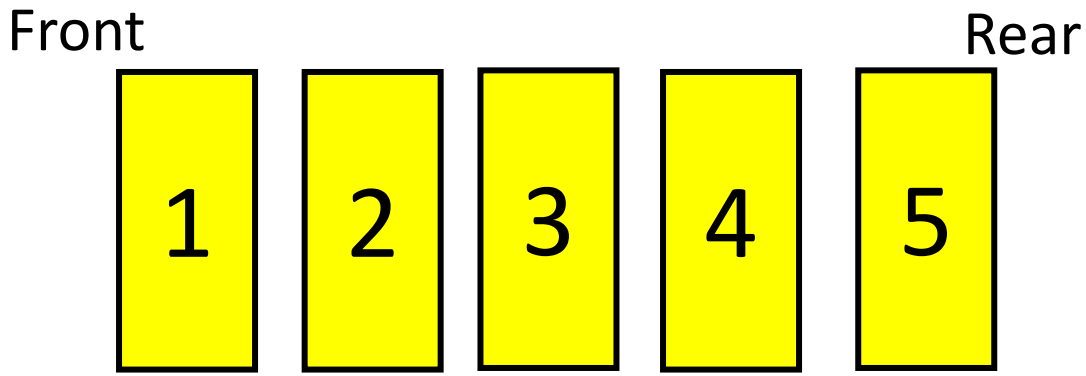
- How the tokens are issued to the customers?

How a person joins or leaves?

What is Queue?

- It is an ordered collections of homogeneous elements.
- Elements are added at the end(tail or rear) of the queue and removed from the front(head) of the queue.
- The first element to be added is the first element to be removed (***FIFO: First In, First Out).***

Queue



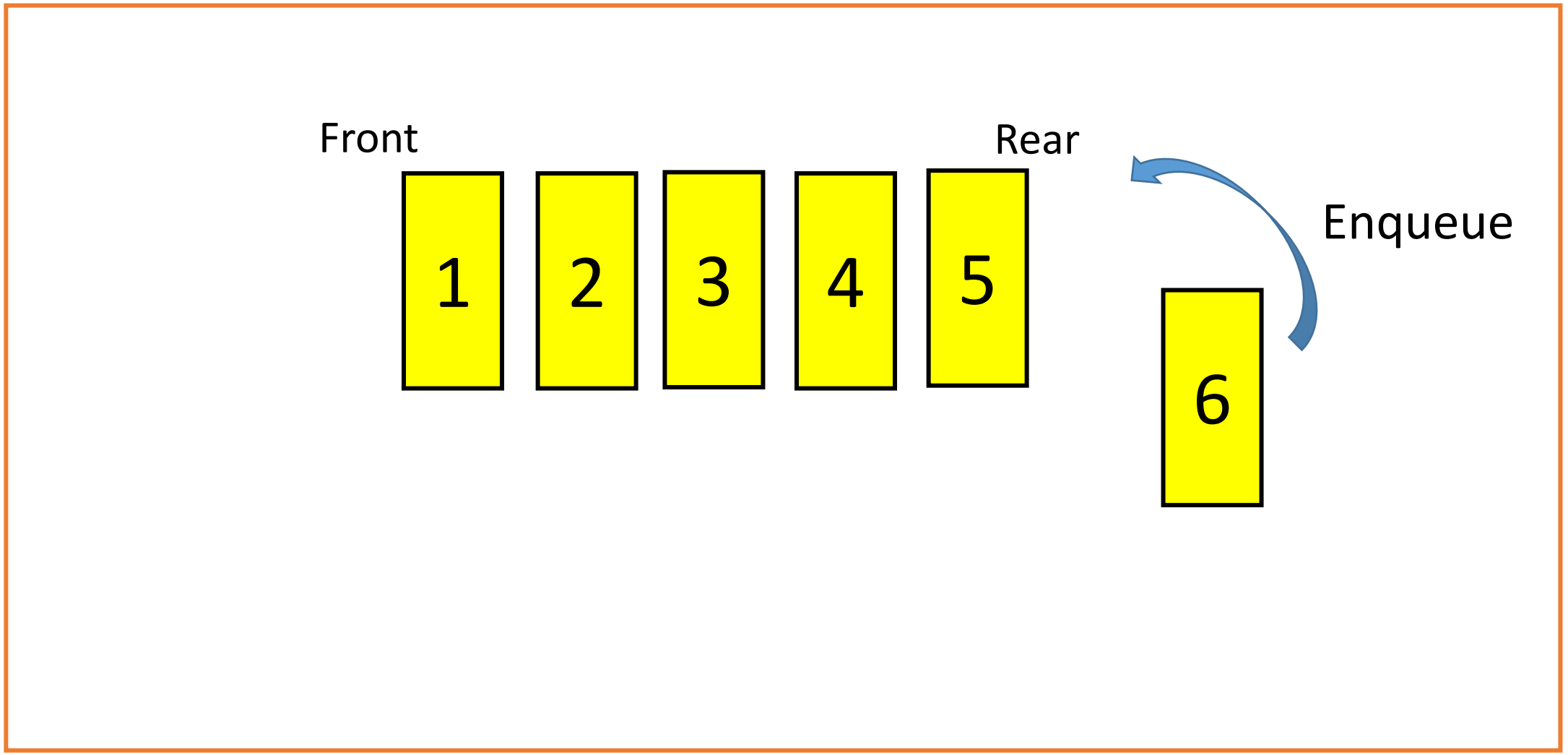
Queue Operations

- Enqueue
- Dequeue
- IsEmpty
- IsFull

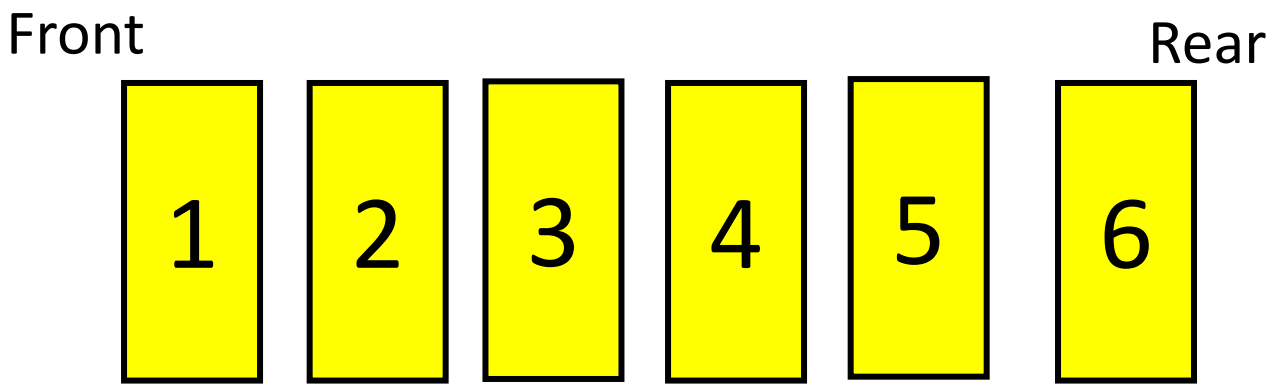
Queue Operations

- **Enqueue:** Adds an element to the rear/back end of queue.
- **Dequeue:** Removes an element from the front end of queue.
- **IsEmpty:** Determines whether the queue is empty.
- **IsFull:** Determines whether the queue is full.

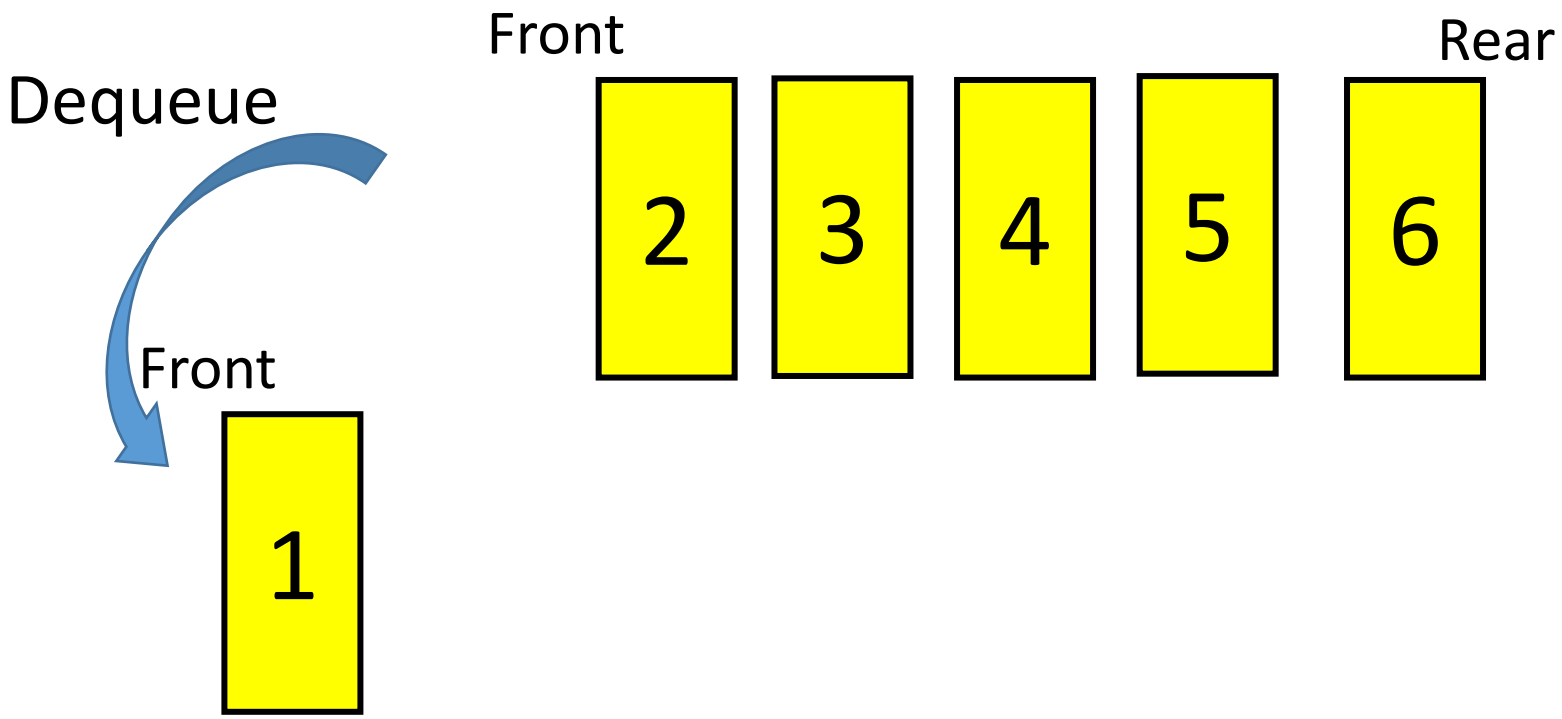
Queue Operations: Enqueue



Queue Operations: Enqueue



Queue Operations: Dequeue



Let watch the animation for better understanding of working of linear Queue…

Queue Operations using arrays

```
struct Queue
{
        int front,rear;
        int elements[MAX];
}queue;

int isfull(queue *);          //To check whether queue is full or not
int isempty(queue *);         //To check whether queue is empty or not
void enqueue(queue *,int);    //To add an element into queue
void dequeue(queue *);        //To delete an element from queue
void display(queue *);        //To display all elements of the queue
```

Queue Operations using arrays

IsFull:

```
 int isfull(queue *pq)
{
        if((pq->front==0)&&(pq->rear==MAX-1))
                return 1;
        else
                return 0;
}
```

Queue Operations using arrays

IsEmpty:

```
 int isempty(queue *pq)
{
        if(pq->front == -1)
                return 1;
        else
                return 0;
}
```

Queue Operations using arrays

Enqueue:

```
 void enqueue(queue *pq,int element)
{       int i;
        if(pq->front==-1)
                pq->front=pq->rear=0;
        else
                pq->rear++;
        pq->elements[pq->rear]=element;
}
```

Queue Operations using arrays

Dequeue:

```
 void dequeue(queue *pq)
{
    int temp;
    temp=pq->elements[pq->front];
    if(pq->front==pq->rear)
        pq->front=pq->rear=-1;
    else
        pq->front++;
    printf("deleted val is %d",temp);
}
```

Queue Operations using arrays

Display:

```
 void display(queue *pq)
{
        int i;
        for(i=pq->front;i<=pq->rear;i++)
        printf("%d\t",pq->elements[i]);
}
```

Let see the code for better understanding of working of linear Queue…

Applications

1. When a resource is shared among multiple consumers. Examples include CPU scheduling, Disk Scheduling.

2. When data is transferred asynchronously (data not necessarily received at same rate as sent) between two processes. Examples include IO Buffers, pipes, file IO, etc.

3. In real life scenario, Call Center phone systems uses Queues to hold people calling them in an order, until a service representative is free.

Examples

1) Implement Toll bay for vehicles.

Note : LMV – 50 Rs, HMV- 100 Rs.

- Read & Display
- Display only HMV vehicle details.
- Count LMV vehicles only.


```
int Count_LMV(queue *pq, char type)
{
        int i,count=0;
        for(i=pq->front;i<=pq->rear;i++)
        {       if(type == 'L')
                count++;
        } return count;
}
```

Home work

- Implement all the functions of queue operations using linked list.

Disadvantages of Linear Queue

KLE TECH

KLE Technological University

Creating Value
Leveraging Knowledge

Circular Queue

KLE TECH

KLE Technological University

Creating Value
Leveraging Knowledge

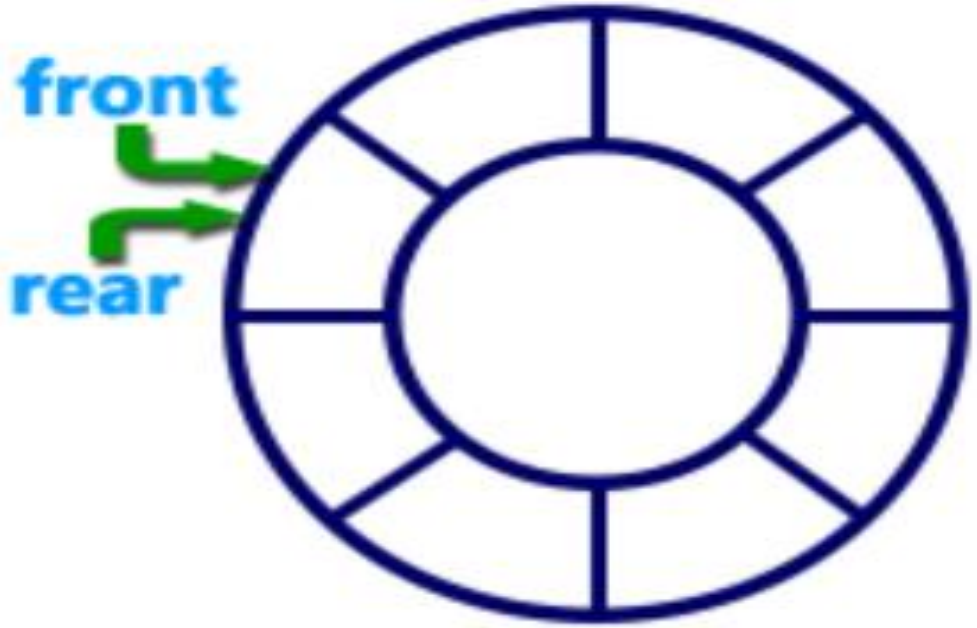
Circular Queues

Circular Queue Examples

- Regular routine of life is an example of circular queue.
  - Eating: Breakfast, Lunch, Snacks, Dinner → Breakfast….
  - Days in a week – Sunday, Monday … Saturday → Sunday ….
  - Months in a year – Jan, Feb … Dec → Jan …

Circular Queue

- Definition: Circular queue is a linear data structure.
- It works based on FIFO (First In First Out) principle. Process the elements in circular manner.



Let watch the animation for better understanding of working of Circular Queue…

Circular Queue Operations

- IsFull
- IsEmpty
- InsertRear
- Deletefront

Circular Queue Operations

- **IsEmpty:** Determines whether the queue is empty.
- **IsFull:** Determines whether the queue is full.
- **InsertRear:** Adds an element to the rear/back end of queue.
- **DeleteFront:** Removes an element from the front end of queue.

Circular Queue Operations

```
typedef struct Q
{
        int rear,front;
        int elements[MAX];
}queue;
```

Circular Queue Operations

```
int isEmpty(queue *pq)
{
        if(pq->rear==-1)
                return(1);
        return(0);
}
```

Circular Queue Operations

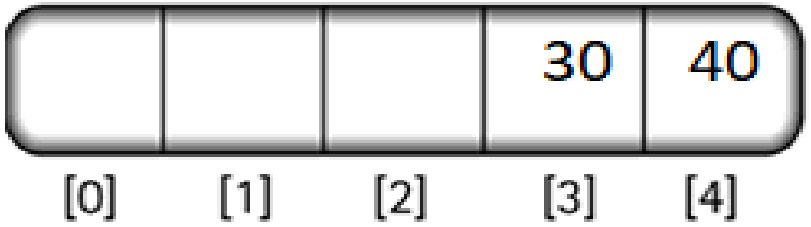
```
int isFull(queue *pq)
{
        if((pq->rear+1)%size==pq->front)
                return(1);
        else
                return(0);
}
```

Circular Queue Operations: Inse

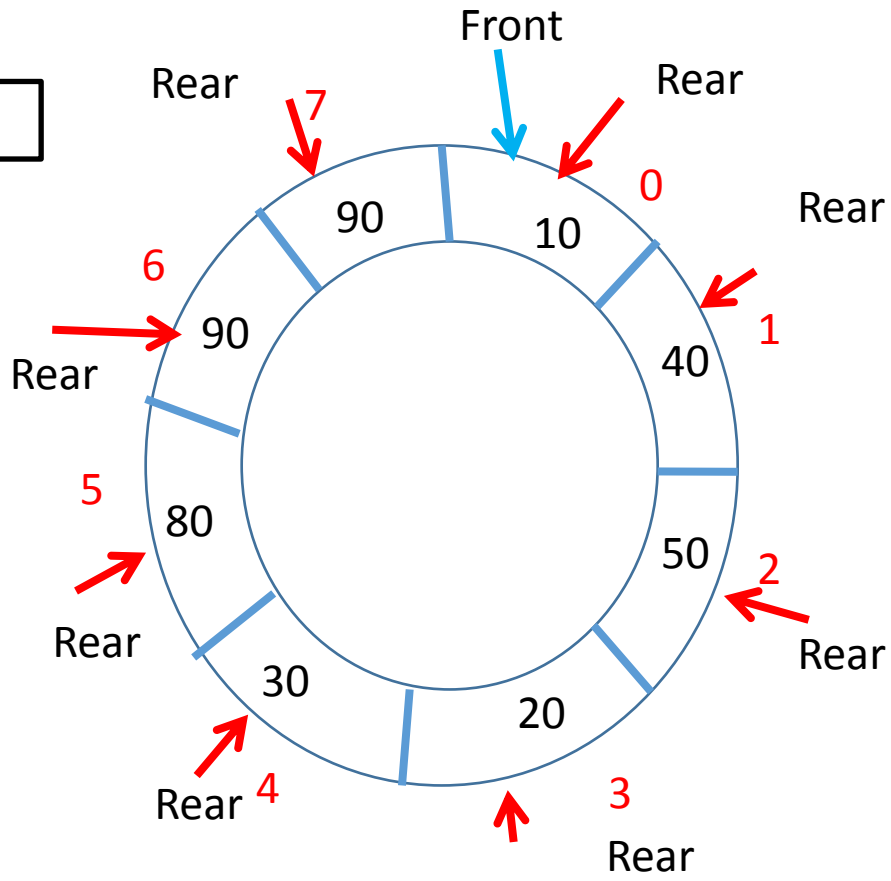
- Rear and Front indices range from 0 to size-1.
- Uses mod operation

Rear=(Rear+1) %Size

(a) Rear is at the bottom of the array




```
front = 3
rear = 4
```



Circular Queue Operations: Insert

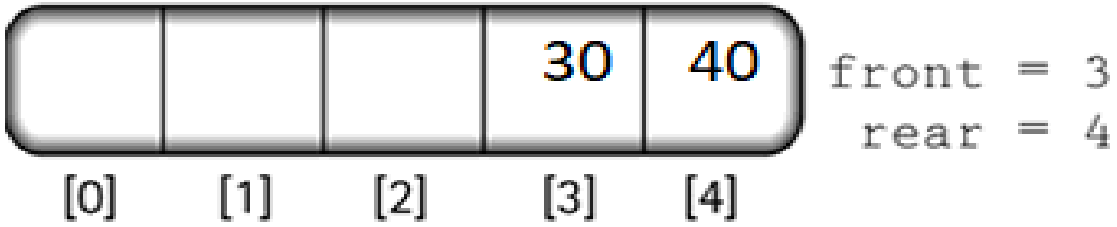

```
void InsertRear(queue *pq, int x)
{
    if(pq->rear==-1)
    {
        pq->rear=pq->front=0;
        pq->elements[pq->rear]=x;
    }
    else
    {
        pq->rear=(pq->rear+1)%Size;
        pq->elements[pq->rear]=x;
    }
}
```

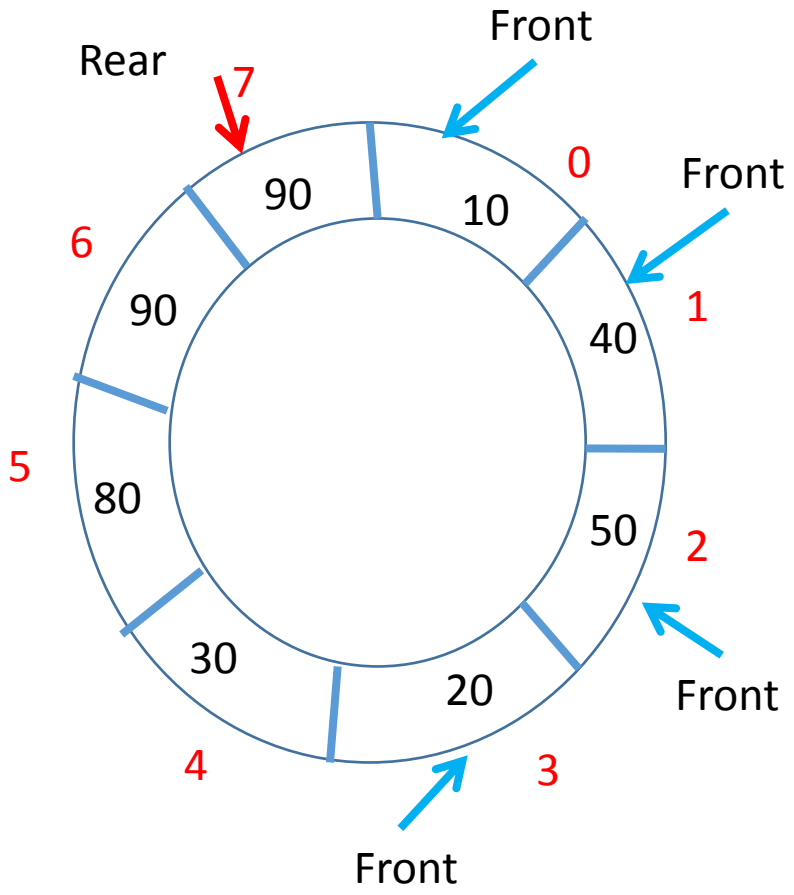
Circular Queue Operations: Delete Front

- Rear and Front indices range from 0 to size-1.
- Uses mod operation

Rear=(Rear+1) %Size

(a) Rear is at the bottom of the array



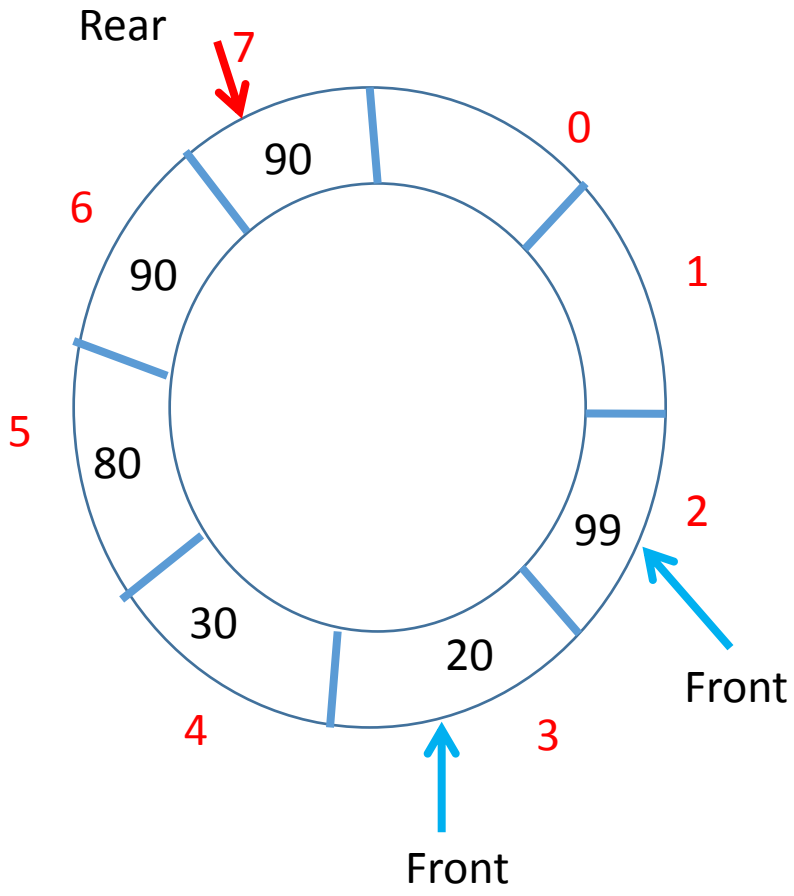


Circular Queue Operations: Delete Front

```
int DeleteFront(queue *pq)
{
    int x;
    x=pq->elements[pq->front];
    if(pq->rear==pq->front)
    {
    pq->rear=-1;
    pq->front=-1;
    }
    else
    pq->front=(pq->front+1)%size;
    return(x);
}
```

Circular Queue Operations

- Adding element after deleting



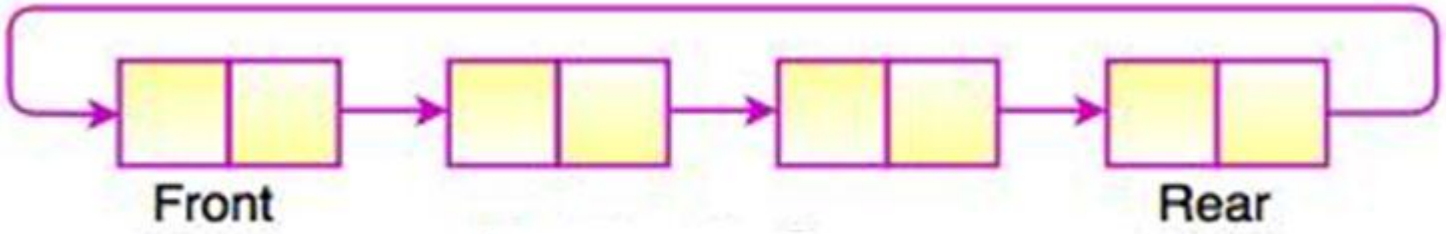
Let see the code for better understanding of working of Circular Queue…

Circular Queue Operations using Linkec

KLE TECH

KLE Technological University
Creating Value
Leveraging Knowledge

- Circular queue is also called as Ring Buffer.
- It can be implemented using circular doubly or singly linked list.



Following functions used for implementation:

Case 1:

Inser_begin(), Delete_end()

Case 2

Insert_end(), Delete_begin()

Circular Queue Applications

- Capping of bottles in Cold-drink company.

- Traffic Signal System

- CPU scheduling and Memory management.
- Buffers Network Routers

KLE TECH

KLE Technological University

Creating Value
Leveraging Knowledge

Home work

- Implement circular queue using linked list.

Next Up..

- Priority Queue
- Double ended Queue
- Multiple Queue