Queue – Introduction

1. Introduction

A **Queue** is a **linear data structure** that follows a specific order for performing operations.

The order followed by a queue is called **FIFO — First In, First Out**.

This means:

- The element inserted first is removed first
- The element inserted last is removed last

Queues are widely used in **scheduling, resource management, and real-time systems**.

---

2. What is a Queue?

A queue is a collection of elements where:

- Insertion happens at one end called the **Rear**
- Deletion happens at the opposite end called the **Front**

Elements move in one direction:

```
Rear → Queue → Front
```

---

3. Real-Life Example of Queue

Common real-world examples of queues:

- People standing in a line at a ticket counter
- Printer job scheduling
- Call waiting systems
- CPU task scheduling

The person who comes first is served first.

---

4. Queue Principle (FIFO)

FIFO (First In, First Out) means:

- The first element added is the first one removed

Example:

```
Enqueue 10
Enqueue 20
Enqueue 30
```

Queue:

```
Front → 10  20  30 ← Rear
```

Dequeue:

```
10 is removed first
```

---

5. Basic Queue Operations

A queue supports the following main operations:

1. **Enqueue** – Add an element at the rear
2. **Dequeue** – Remove an element from the front
3. **Front / Peek** – View the front element
4. **isEmpty** – Check if the queue is empty
5. **isFull** – Check if the queue is full (array queue)

---

6. How Queue Works (Plain English Logic)

1. Queue starts empty

2. New elements are added at the rear
3. Elements are removed from the front
4. Front and rear pointers update after every operation
5. No random access is allowed

---

7. Queue Representation

Queues can be implemented using:

- **Array**
- **Linked List**

Both implementations follow the FIFO principle.

---

8. Queue Overflow and Underflow

Queue Overflow

- Occurs when trying to add an element to a full queue

Queue Underflow

- Occurs when trying to remove an element from an empty queue

These are common queue error conditions.

---

9. Applications of Queue

Queues are used in:

- CPU scheduling
- Disk scheduling
- Breadth-First Search (BFS)
- Printer queues
- Data buffering

- Network packet handling

10. Advantages of Queue

- Simple and efficient structure
- Maintains order of processing
- Ideal for scheduling tasks
- Easy to implement

11. Limitations of Queue

- No random access
- Fixed size in array implementation
- Inefficient memory usage in simple array queues

12. Time Complexity of Queue Operations

Operation	Time Complexity
Enqueue	O(1)
Dequeue	O(1)
Peek	O(1)

All queue operations execute in **constant time**.

13. Summary

- Queue is a linear data structure
- Follows FIFO principle
- Insertion at rear, deletion at front
- Widely used in scheduling systems
- Simple and efficient