📘Teaching Notes: Queue (Unit 2 – Part 1)

---

1. Introduction to Queue

Definition: A Queue is a linear data structure that follows FIFO (First-In-First-Out) principle.

- The first element inserted is the first one to be removed.

Characteristics of Queue:

- Insertion happens at the rear.
- Deletion happens at the front.
- Follows FIFO order.

Real-life Examples:

- People standing in a line at a ticket counter.
- Print jobs sent to a printer.
- CPU task scheduling.

Comparison with Stack:

Feature	Stack	Queue
Order	LIFO (Last In First Out)	FIFO (First In First Out)
Insertion	Top	Rear
Deletion	Top	Front
Example	Book stack	Ticket counter queue

---

1. Abstract Data Type (ADT) of Queue

Queue ADT defines the operations that can be performed on a queue.

Queue Operations:

Operation	Description
enqueue()	Add an element at the rear of the queue
dequeue()	Remove an element from the front
peek()/front()	View the front element without removing it
isEmpty()	Check if queue has no elements
isFull()	Check if queue has reached its maximum size

Example of ADT operations in C (conceptual):

```
struct Queue {
    int front, rear, size;
    unsigned capacity;
    int* array;
};
```

---

1. Array Implementation of Queue

Representation using Array:

- A queue can be implemented using a fixed-size array.
- Two pointers are used:
- front → points to the first element
- rear → points to the last element

Queue Operations in Array

C Program Example:

```
#include <stdio.h>
#define MAX 5

int queue[MAX];
int front = -1, rear = -1;

int isEmpty() {
    return front == -1;
}

int isFull() {
    return rear == MAX - 1;
}

void enqueue(int value) {
    if(isFull()) {
        printf("Queue Overflow\n");
        return;
    }
    if(front == -1) front = 0;
    rear++;
    queue[rear] = value;
    printf("%d enqueued to queue\n", value);
}

void dequeue() {
    if(isEmpty()) {
        printf("Queue Underflow\n");
        return;
    }
    printf("%d dequeued from queue\n", queue[front]);
    front++;
    if(front > rear) front = rear = -1;
}

void display() {
    if(isEmpty()) {
        printf("Queue is empty\n");
        return;
    }
    printf("Queue elements: ");
    for(int i = front; i <= rear; i++)
        printf("%d ", queue[i]);
    printf("\n");
}

int main() {
    enqueue(10);
    enqueue(20);
    enqueue(30);
    display();

    dequeue();
    display();

    enqueue(40);
    enqueue(50);
    enqueue(60);
    display();

    return 0;
}
```

Output Example:

```
10 enqueued to queue
20 enqueued to queue
30 enqueued to queue
Queue elements: 10 20 30
10 dequeued from queue
Queue elements: 20 30
40 enqueued to queue
50 enqueued to queue
Queue Overflow
Queue elements: 20 30 40 50
```

Limitations of Simple Array Queue:

1. Fixed size – cannot exceed array size.
2. Unused space problem – space at beginning wasted after dequeues.

Overflow & Underflow Conditions:

- Overflow: Attempt to enqueue when the queue is full.
- Underflow: Attempt to dequeue when the queue is empty.