INFORMATIKA UMN

UMN
UNIVERSITAS
MULTIMEDIA
NUSANTARA

IF232
ALGORITHMS
&
DATA STRUCTURES

07
QUEUES

DENNIS GUNAWAN

REVIEW

Stacks:

Array Representation of Stacks

Operations on a Stack

Linked Representation of Stacks

Operations on a Linked Stack

Applications of Stacks

OUTLINE

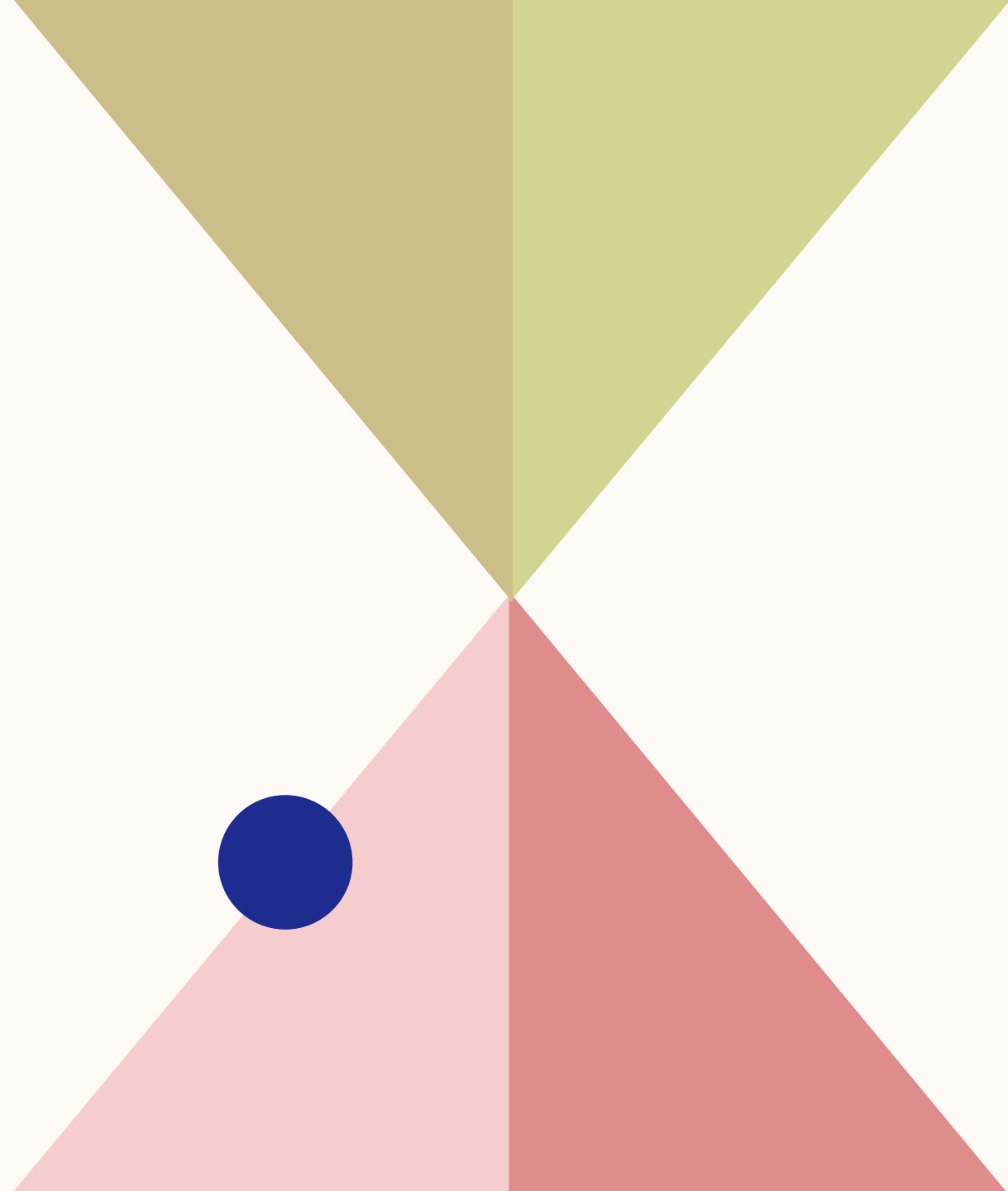
Array Representation of Queues

Operations on Queues

Linked Representation of Queues

Operations on Linked Queues

Applications of Queues



QUEUES

- People moving on an escalator
  - The people who got on the escalator first will be the first one to step out of it
- People waiting for a bus
  - The first person standing in the line will be the first one to get into the bus
- People standing outside the ticketing window of a cinema hall
  - The first person in the line will get the ticket first and thus will be the first one to move out of it

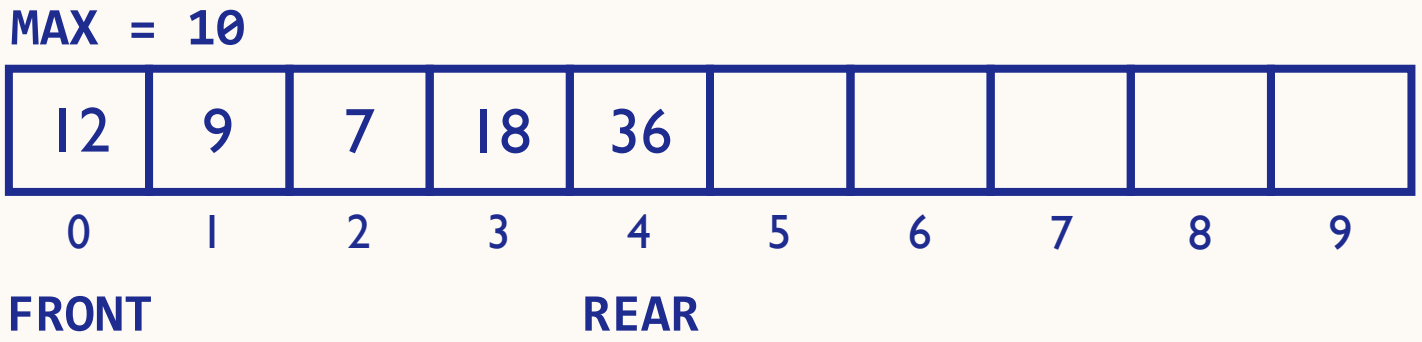
QUEUES

- Luggage kept on conveyor belts
  - The bag which was placed first will be the first to come out at the other end
- Cars lined at a toll bridge
  - The first car to reach the bridge will be the first to leave

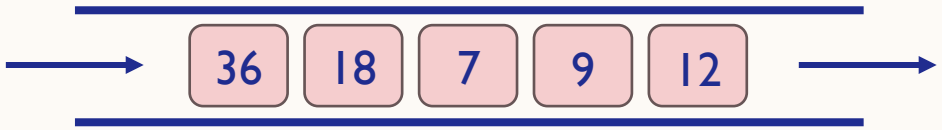
QUEUES

- FIFO: First In First Out
- The elements in a queue are added at one end called the **REAR** and removed from the other end called the **FRONT**
- **MAX** is the size of the queue

ARRAY REPRESENTATION OF QUEUES

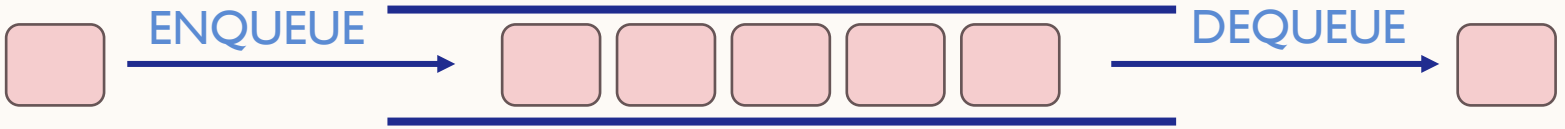


- The queue is empty: **FRONT = -1** and **REAR = -1** **FRONT = REAR + 1**
- The queue is full: **REAR = MAX-1**



OPERATIONS ON QUEUES

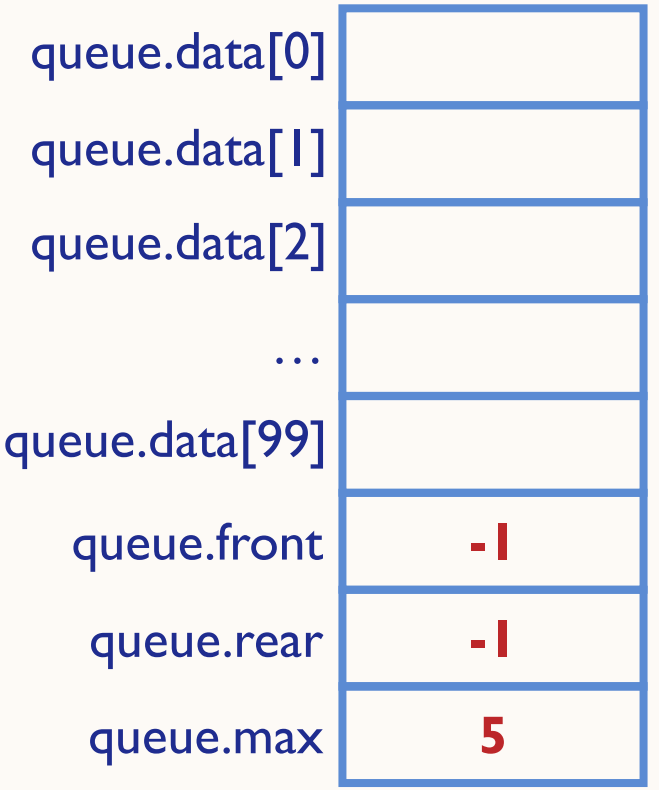
- **Enqueue**: inserts an element in the queue
- **Dequeue**: deletes an element from the queue
- **Peek**: returns the value of the frontmost element of the queue



DECLARATION

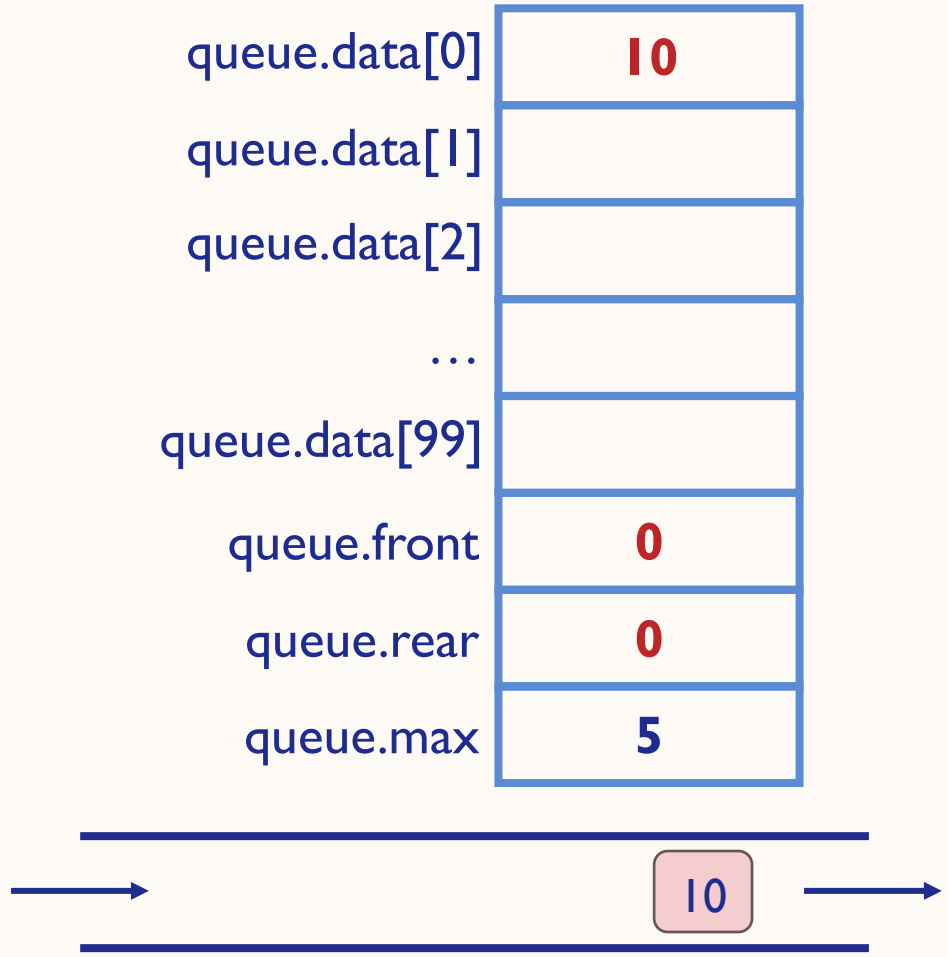
```
struct tqueue{
    int data[100];
    int front;
    int rear;
    int max;
};
```

```
int main()
{
    struct tqueue queue;
    int number;
    ...
    queue.front = -1;
    queue.rear = -1;
    queue.max = 5;
    ...
}
```



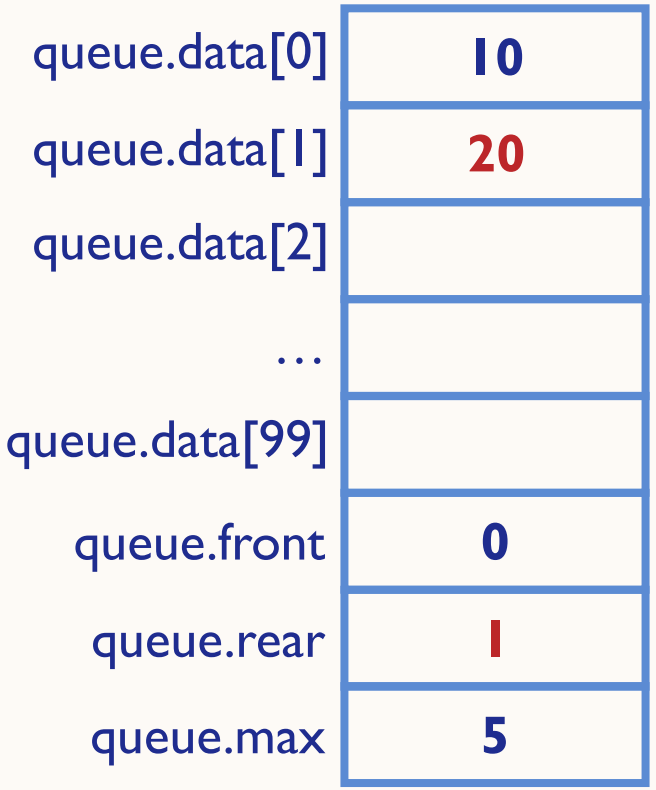
ENQUEUE

```
scanf("%d", &number); //10

if(queue.rear == queue.max - 1)
    printf("OVERFLOW");
else if(queue.front == -1 && queue.rear == -1)
    queue.front = queue.rear = 0;
else
    queue.rear++;
queue.data[queue.rear] = number;
```



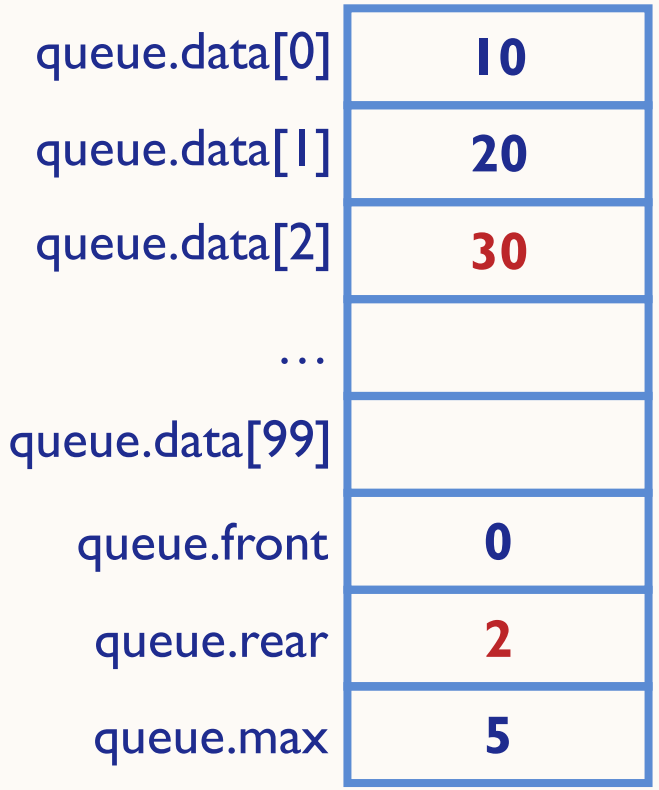
ENQUEUE

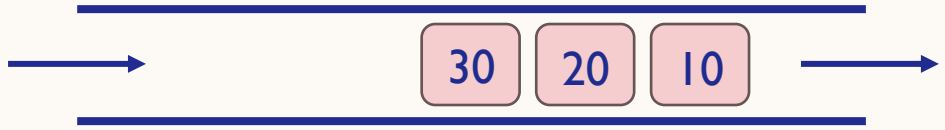
```
scanf("%d", &number); //20

if(queue.rear == queue.max - 1)
    printf("OVERFLOW");
else if(queue.front == -1 && queue.rear == -1)
    queue.front = queue.rear = 0;
else
    queue.rear++;
queue.data[queue.rear] = number;
```



ENQUEUE

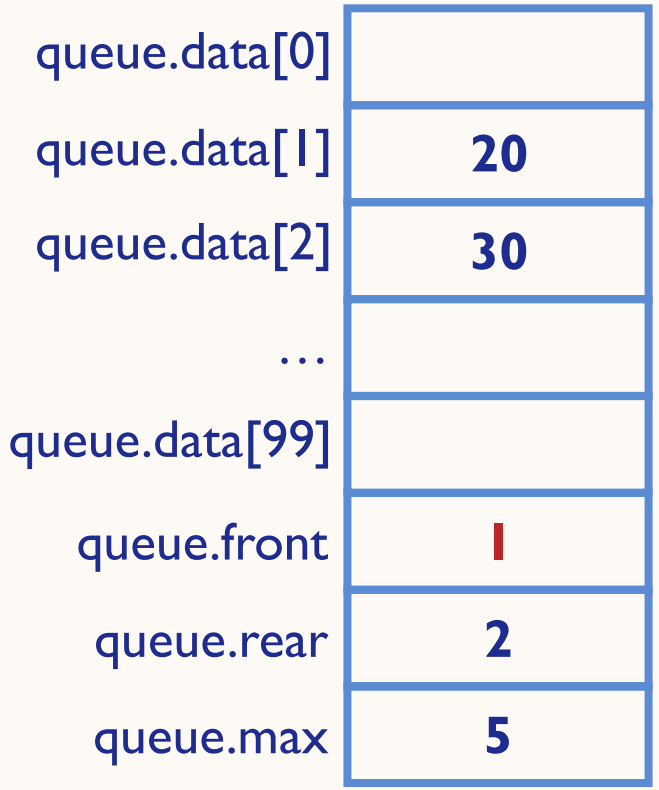
```
scanf("%d", &number); //30

if(queue.rear == queue.max - 1)
    printf("OVERFLOW");
else if(queue.front == -1 && queue.rear == -1)
    queue.front = queue.rear = 0;
else
    queue.rear++;
queue.data[queue.rear] = number;
```





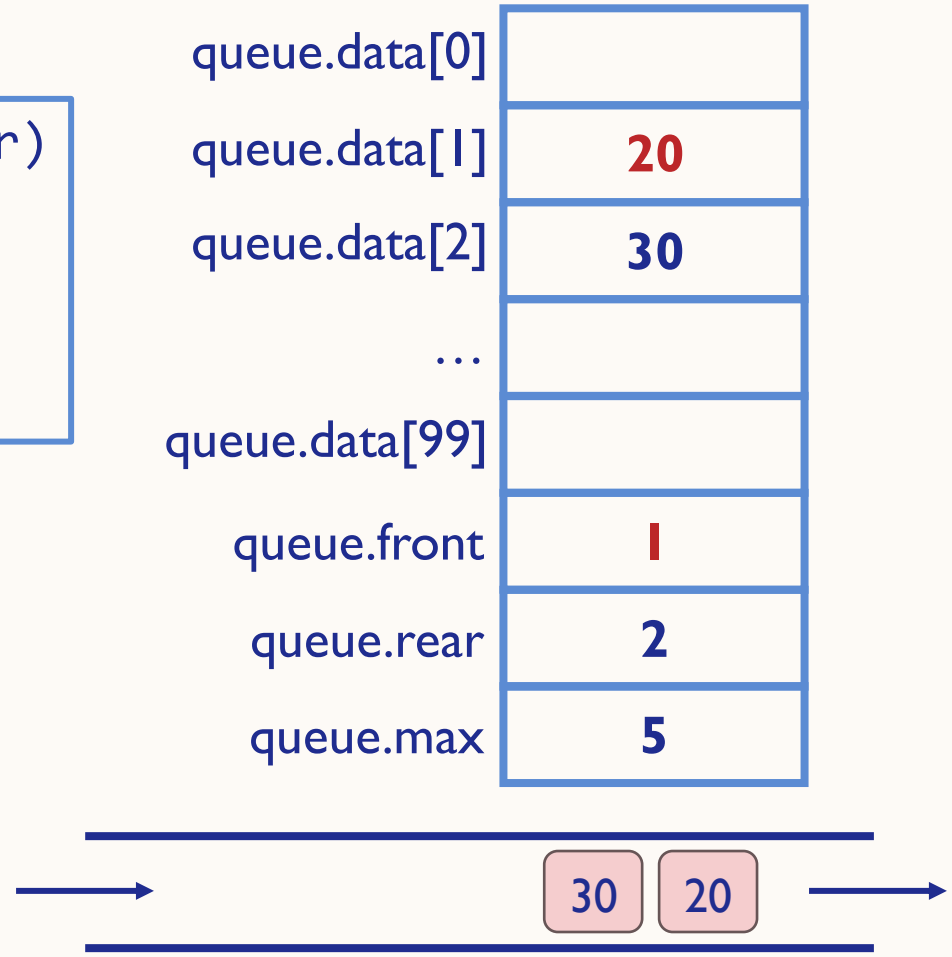
DEQUEUE

```
if(queue.front == -1 || queue.front > queue.rear)
    printf("UNDERFLOW");
else{
    number = queue.data[queue.front]; //10
    queue.front++;
    if(queue.front > queue.rear)
        queue.front = queue.rear = -1;
}
```



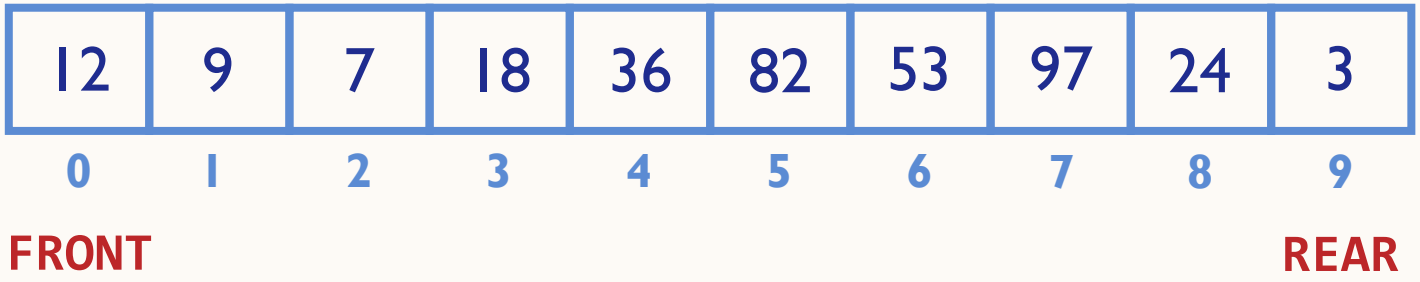
PEEK

```
if(queue.front == -1 || queue.front > queue.rear)
    printf("EMPTY");
else{
    //process queue.data[queue.front] 20
}
```

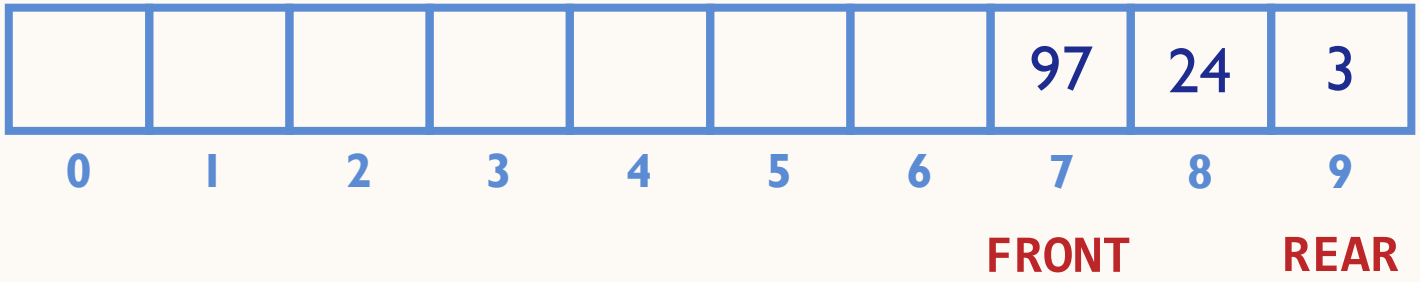


FULL?

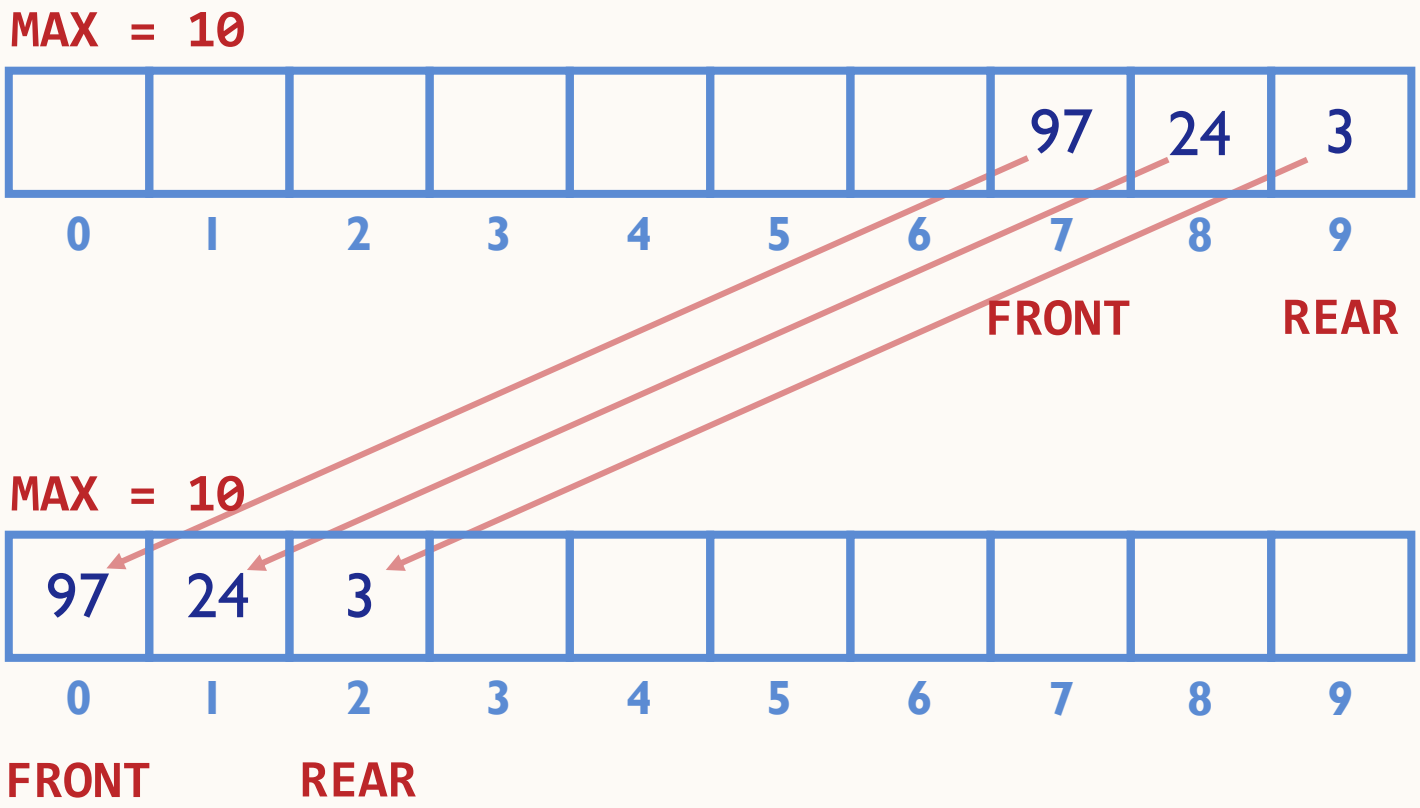
MAX = 10



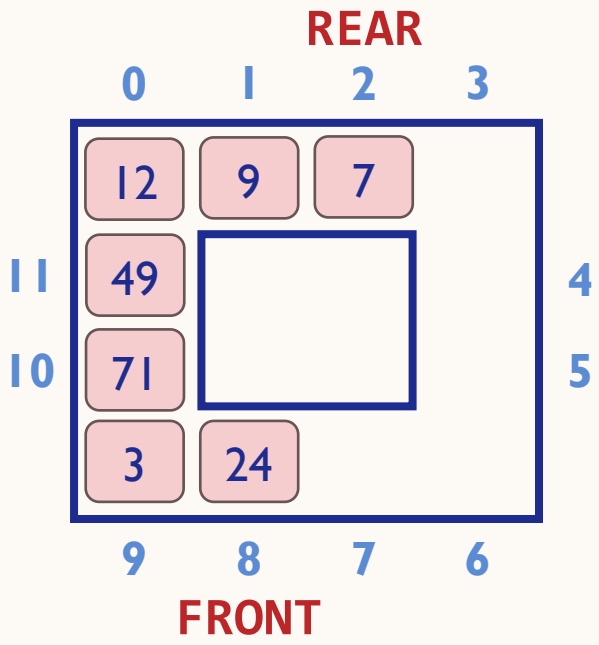
MAX = 10



SHIFT THE ELEMENTS



CIRCULAR QUEUE



SIZE = 12 MAX = SIZE = 12	SIZE = 11 MAX = SIZE + 1 = 12
The queue is empty: FRONT = -1 and REAR = -1	The queue is empty: FRONT = (REAR + 1) % MAX
The queue is full: FRONT = (REAR + 1) % MAX	The queue is full: FRONT = (REAR + 2) % MAX

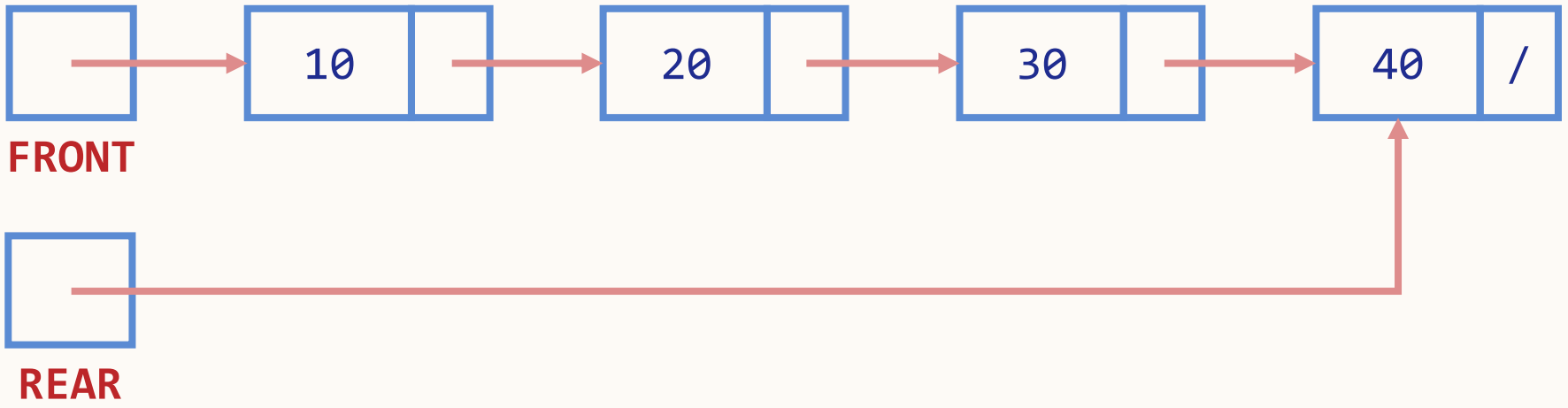
REAR for enqueue:

```
REAR = (REAR + 1) % MAX
```

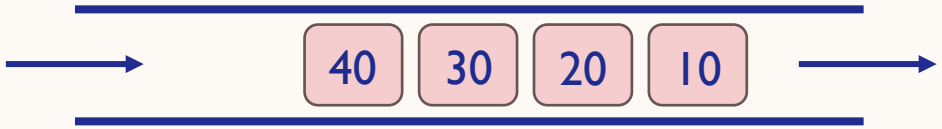
FRONT after dequeue:

```
FRONT = (FRONT + 1) % MAX
```

LINKED REPRESENTATION OF QUEUES



- All insertions will be done at the **REAR** end and all the deletions will be done at the **FRONT** end
- The queue is empty: **FRONT = REAR = NULL**

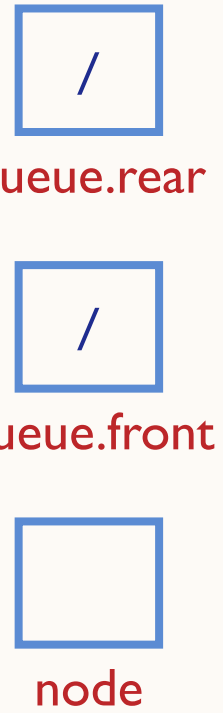


DECLARATION

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

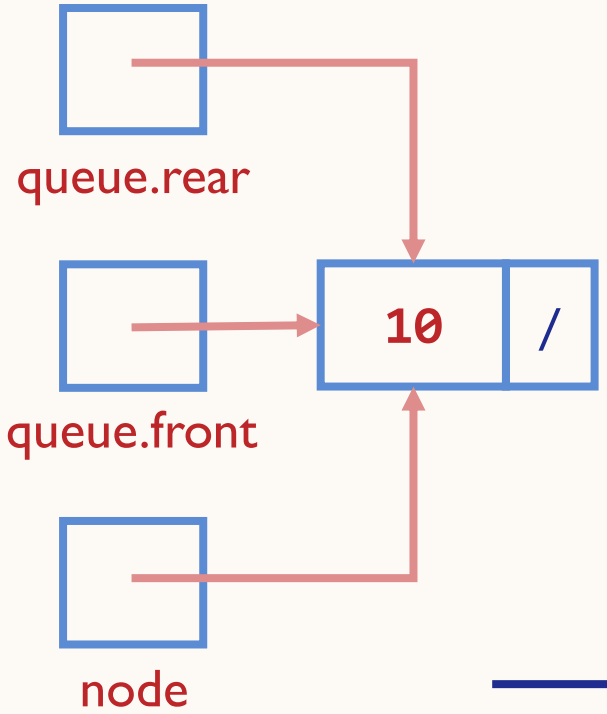
```
struct tqueue{
    struct tnode *front;
    struct tnode *rear;
};
```

```
int main()
{
    struct tqueue queue;
    struct tnode *node;
    int number;
    ...
    queue.front = NULL;
    queue.rear = NULL;
    ...
}
```



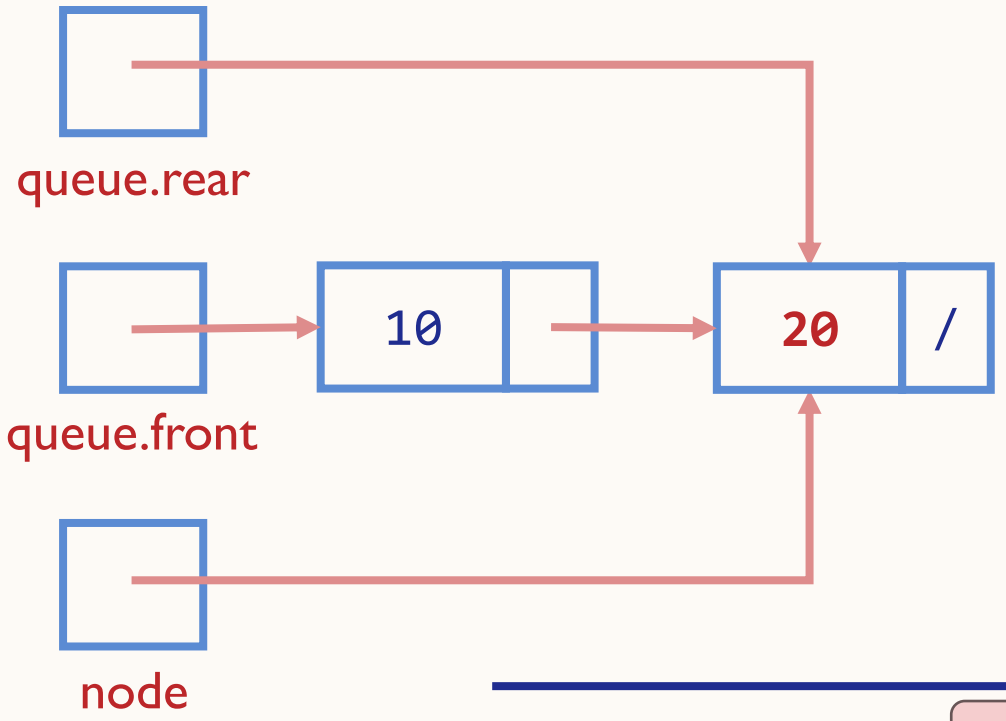
ENQUEUE

```
scanf("%d", &number); //10

node = (struct tnode *) malloc
       (sizeof(struct tnode));
node->data = number;
node->next = NULL;

if(queue.front == NULL)
   queue.front = node;
else
   queue.rear->next = node;
queue.rear = node;
```



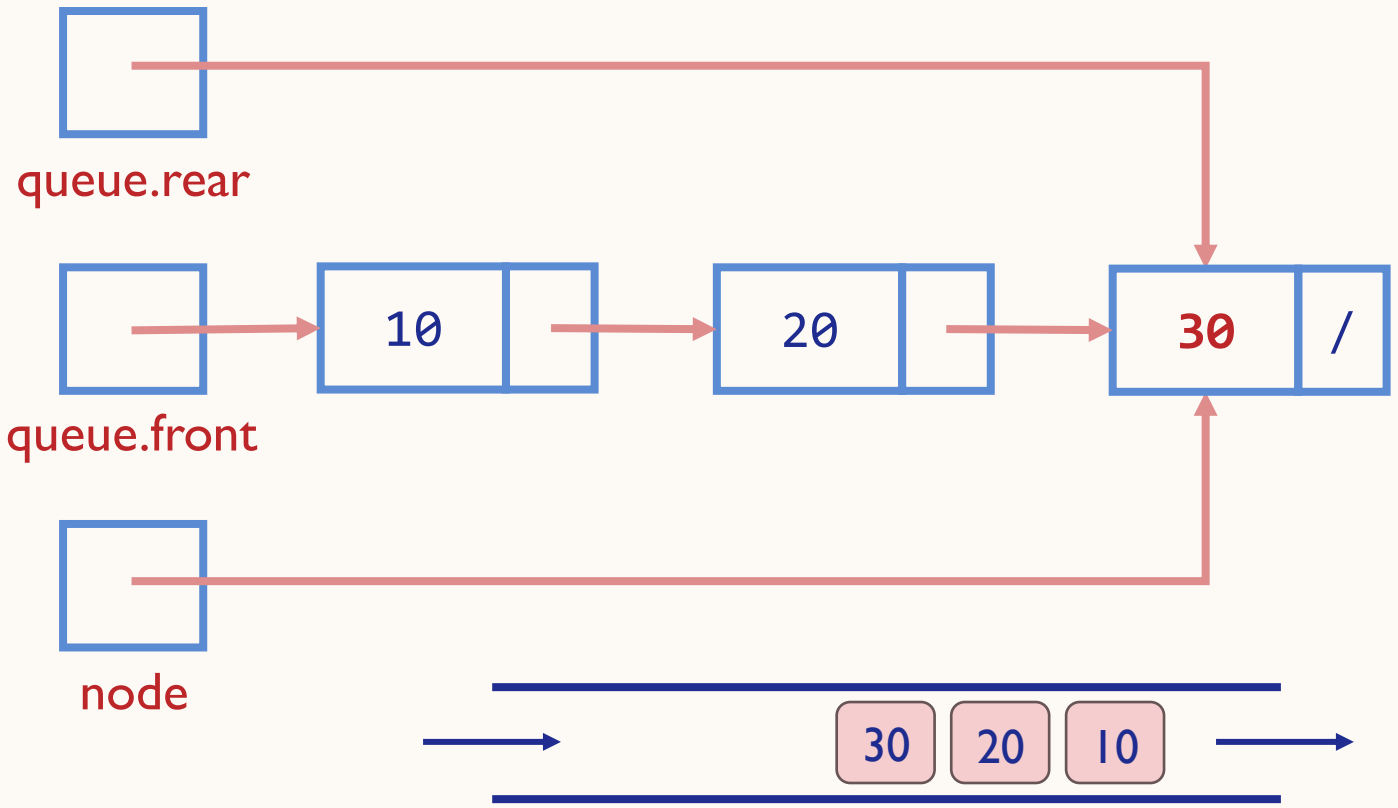
ENQUEUE

```
scanf("%d", &number); //20

node = (struct tnode *) malloc
       (sizeof(struct tnode));
node->data = number;
node->next = NULL;

if(queue.front == NULL)
   queue.front = node;
else
   queue.rear->next = node;
queue.rear = node;
```



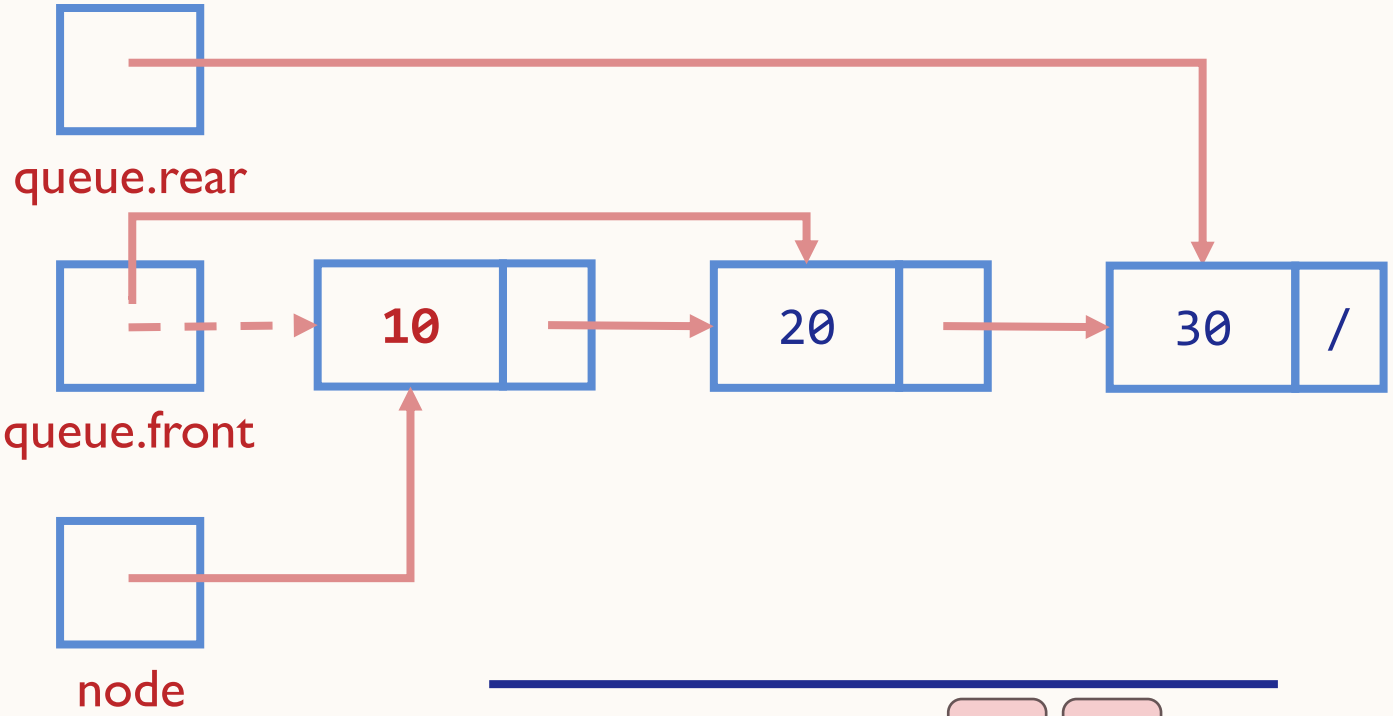
ENQUEUE

```
scanf("%d", &number); //30

node = (struct tnode *) malloc
       (sizeof(struct tnode));
node->data = number;
node->next = NULL;

if(queue.front == NULL)
   queue.front = node;
else
   queue.rear->next = node;
queue.rear = node;
```



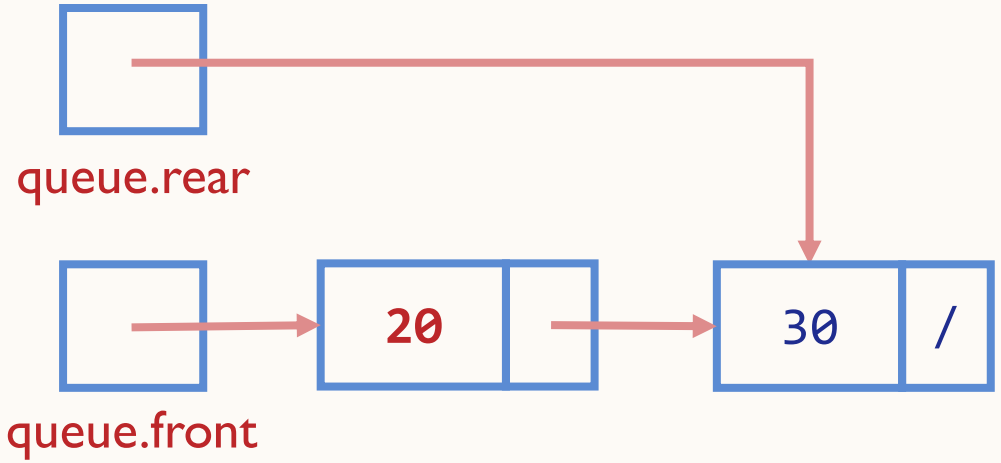
DEQUEUE

```
if(queue.front == NULL)
    printf("UNDERFLOW");
else{
    node = queue.front;
    queue.front = queue.front->next;
    //process node->data 10
    free(node);
}
```



PEEK

```
if(queue.front == NULL)
    printf("EMPTY");
else{
    //process queue.front->data 20
}
```



APPLICATIONS OF QUEUES

- Waiting lists for a single shared resource like printer, disk, CPU
- Transfer data asynchronously (data not necessarily received at same rate as sent) between two processes (IO buffers), e.g., pipes, file IO, sockets
- Buffers on MP3 players
- Used in playlist for jukebox to add songs to the end, play from the front of the list
- Used in operating system for handling interrupts

PRACTICE

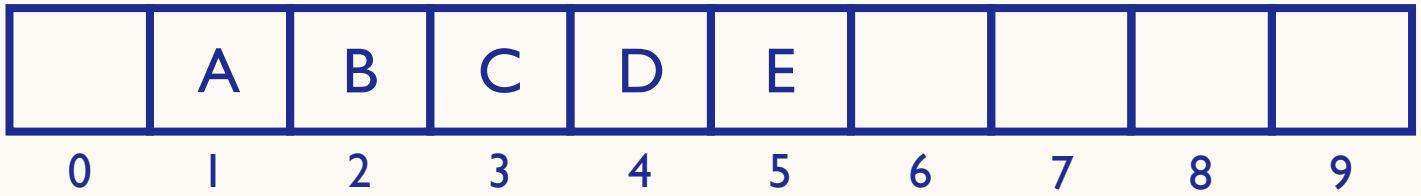
EXERCISES

1. Draw the queue structure in each case when the following operations are performed on an empty queue.

    a. Add A, B, C, D, E, F
    b. Delete two letters
    c. Add G
    d. Add H
    e. Delete four letters
    f. Add I

EXERCISES

2. Consider the queue given below which has **FRONT = 1** and **REAR = 5**.



Now perform the following operations on the queue:

a. Add F
b. Delete two letters
c. Add G
d. Add H
e. Delete four letters
f. Add I

EXERCISES

3. Write a formula to calculate the number of items in each of the following queues.

    a. Linear queue
    b. Circular queue

REFERENCES

- Deitel, P. and Harvey Deitel (2022), C How to Program (9th Edition), Pearson Education.
- Thareja, R. (2014), Data Structures Using C (2nd Edition), India: Oxford University Press.

NEXT

Trees:

Basic Terminology

Types of Trees

Traversing a Binary Tree

Applications of Trees

VISION

To become an **outstanding** undergraduate Computer Science program that produces **international-minded** graduates who are **competent** in software engineering and have **entrepreneurial spirit** and **noble character**.

MISSION

1. To conduct studies with the best technology and curriculum, supported by professional lecturer
2. To conduct research in Informatics to promote science and technology
3. To deliver science-and-technology-based society services to implement science and technology

Without hard work,
nothing grows but weeds.

UMN
UNIVERSITAS MULTIMEDIA NUSANTARA

INFORMATIKA UMN

Have patience.
All things are difficult before they become easy.