Queue

INTRODUCTION

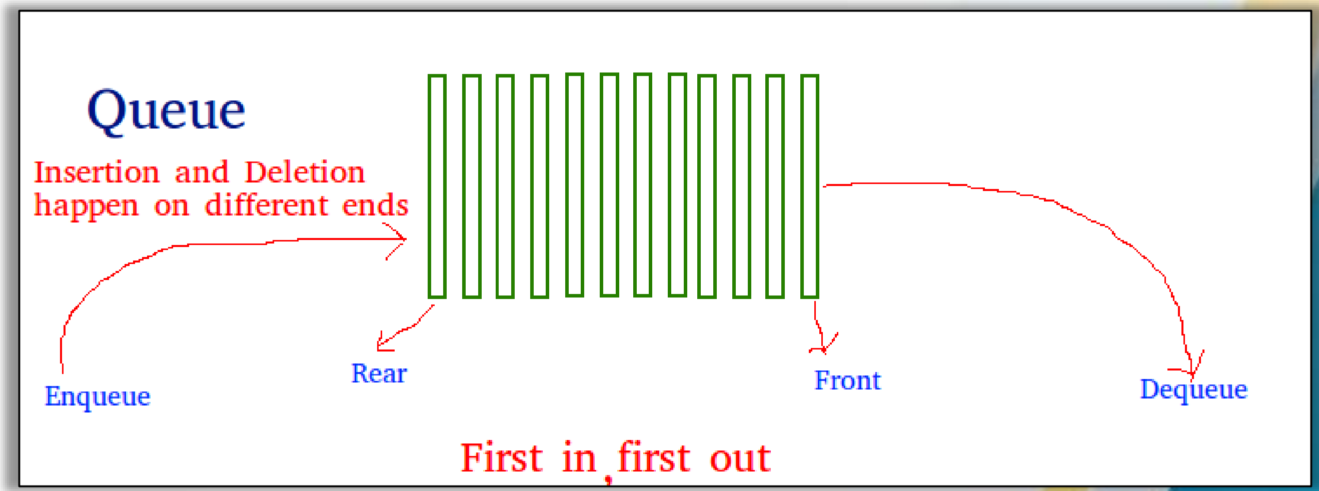
- This segment requires that you have a solid understanding of arrays and have studied Exceptions and interfaces.

- This will develop your programming skills and have strong understanding of stacks and queues and strengthen your abilities in working with arrays, develop a moderate facility with linked lists, and learn to use recursion.

What is Queue?

What is Queue?

- **A**n abstract data type or a linear data structure, just like stack data structure, in which the first element is inserted from one end called the **REAR**(*also called **tail***), and the removal of existing element takes place from the other end called as **FRONT**(*also called **head***).
- **First-In-First-Out** methodology, i.e., the data item stored first will be accessed first.



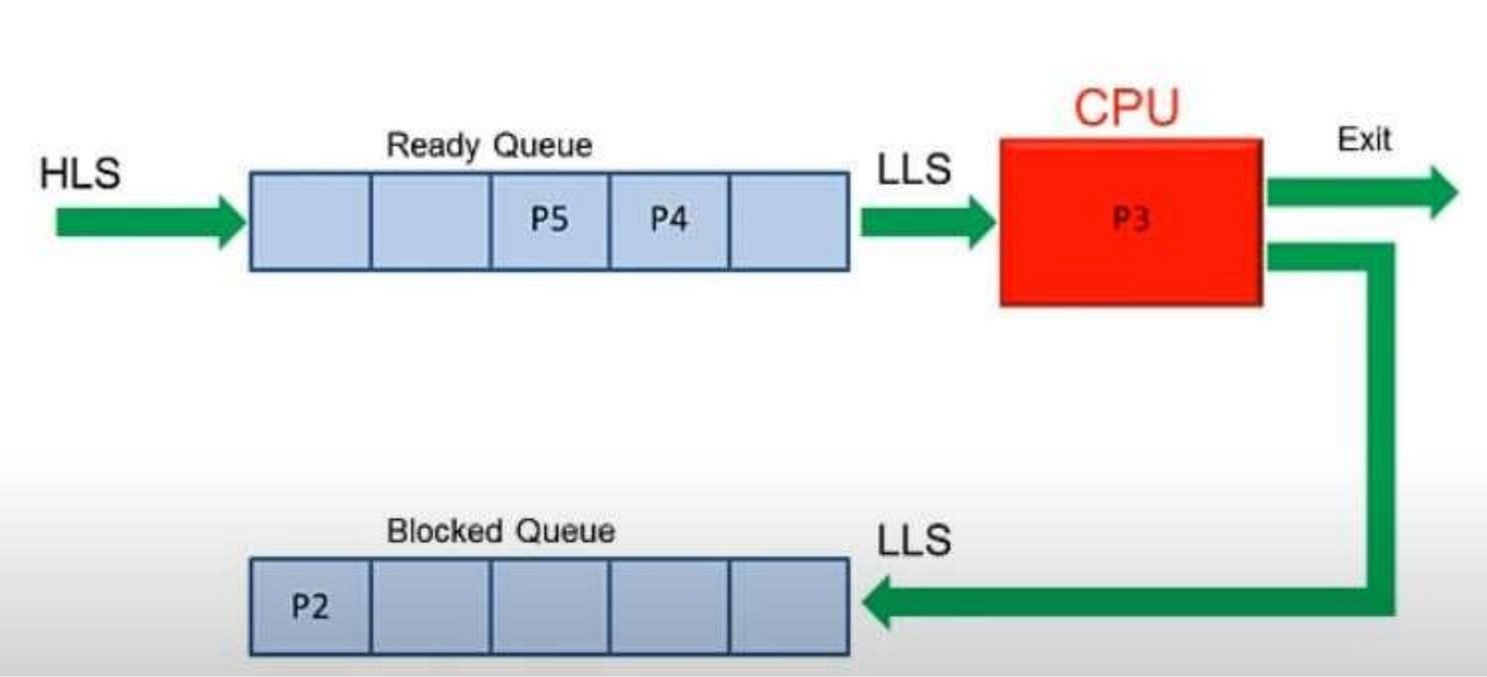
Real life examples of queue are:

- **A queue of people at ticket-window:** The person who comes first gets the ticket first. The person who is coming last is getting the tickets in last. ...

- **Vehicles on toll-tax bridge:** The vehicle that comes first to the toll tax booth leaves the booth first.

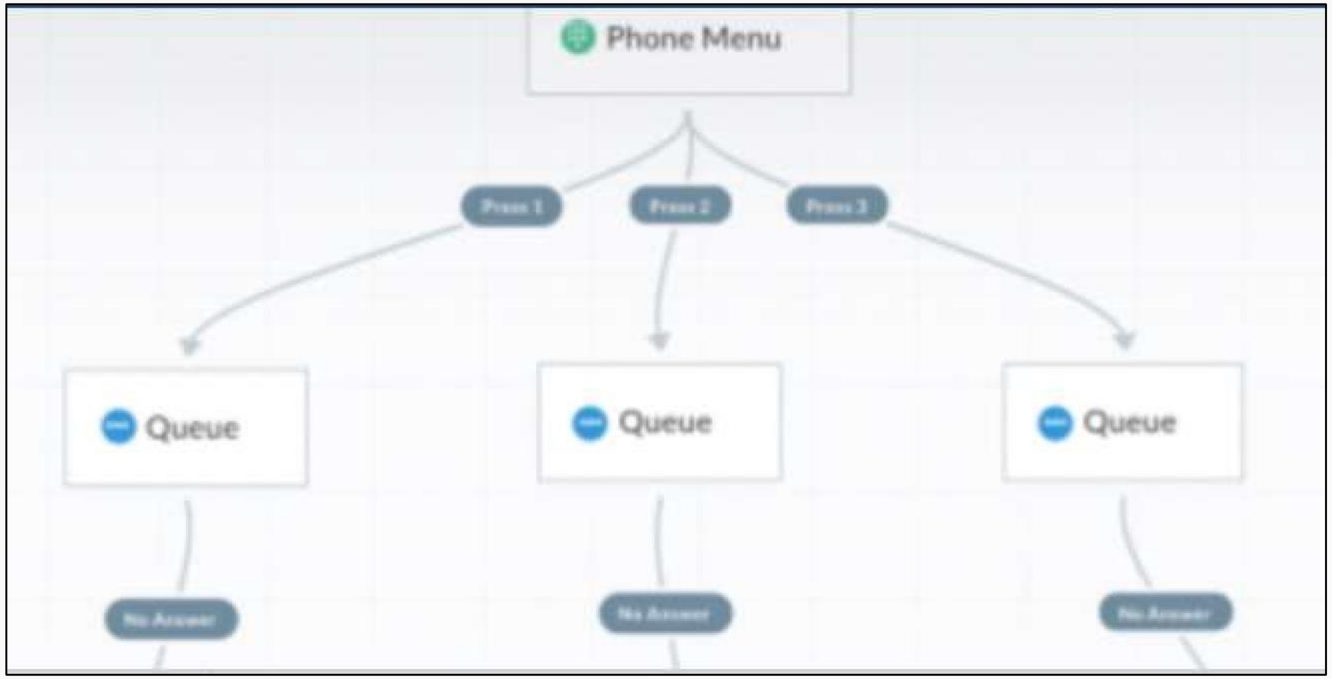
Applications of Queue (Computer)

1. Serving requests on a single shared resource, like a printer, CPU task scheduling etc.



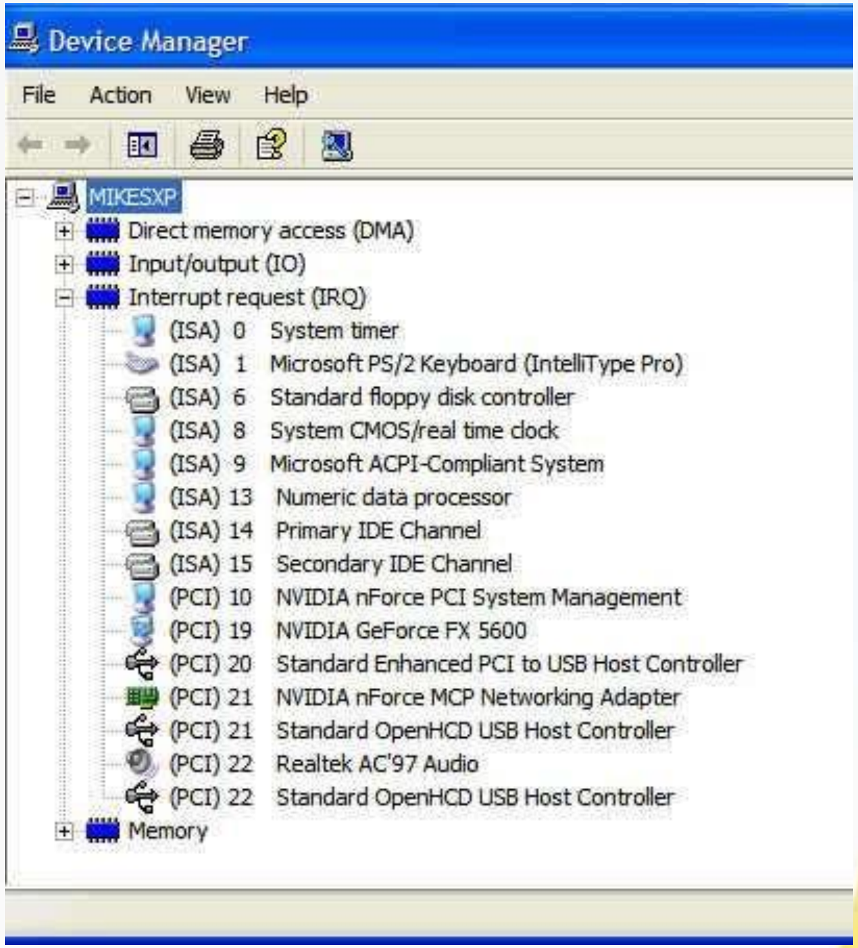
Applications of Queue

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



Applications of Queue

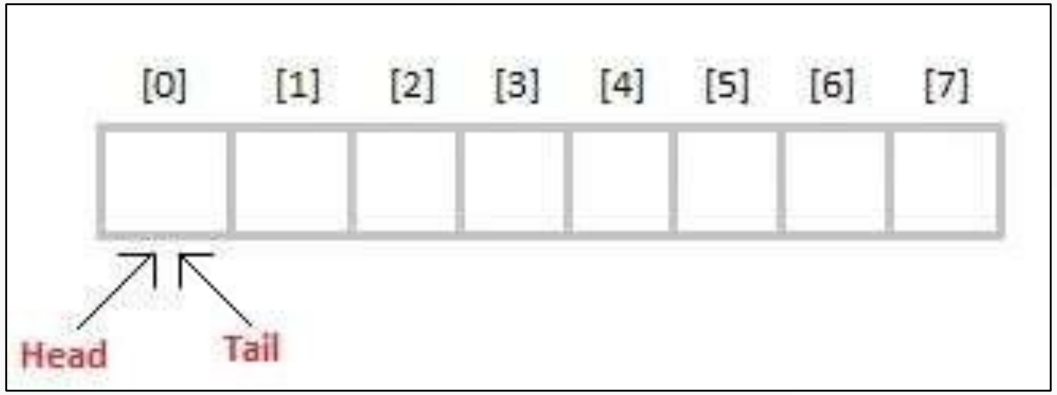
3. Handling of interrupts in real-time systems. The interrupts are handled in the same order as they arrive i.e **First come first served.**



```
MIKESXP
  Direct memory access (DMA)
  Input/output (IO)
  Interrupt request (IRQ)
    (ISA)  0   System timer
    (ISA)  1   Microsoft PS/2 Keyboard (IntelliType Pro)
    (ISA)  6   Standard floppy disk controller
    (ISA)  8   System CMOS/real time clock
    (ISA)  9   Microsoft ACPI-Compliant System
    (ISA) 13   Numeric data processor
    (ISA) 14   Primary IDE Channel
    (ISA) 15   Secondary IDE Channel
    (PCI) 10   NVIDIA nForce PCI System Management
    (PCI) 19   NVIDIA GeForce FX 5600
    (PCI) 20   Standard Enhanced PCI to USB Host Controller
    (PCI) 21   NVIDIA nForce MCP Networking Adapter
    (PCI) 21   Standard OpenHCD USB Host Controller
    (PCI) 22   Realtek AC'97 Audio
    (PCI) 22   Standard OpenHCD USB Host Controller
  Memory
```

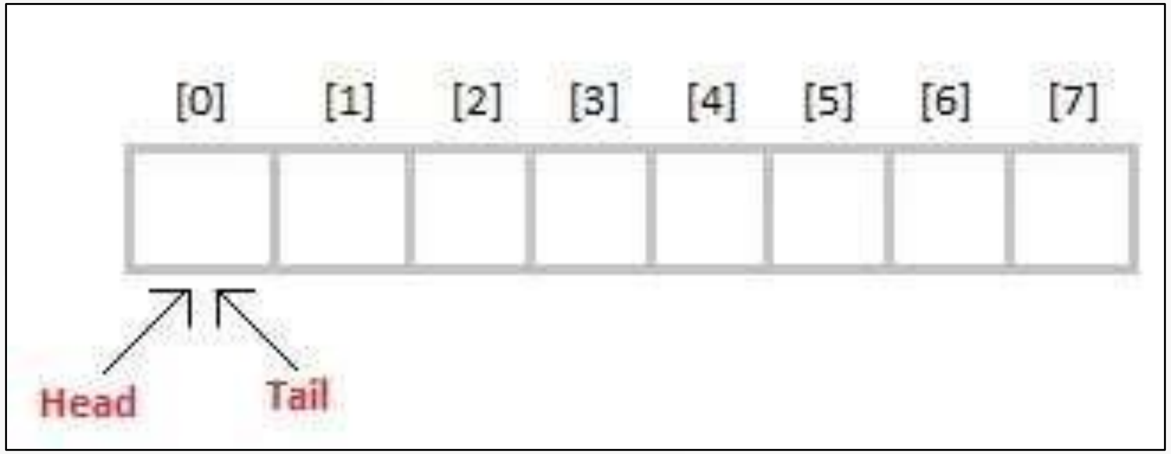
Implementation of Queue Data Structure

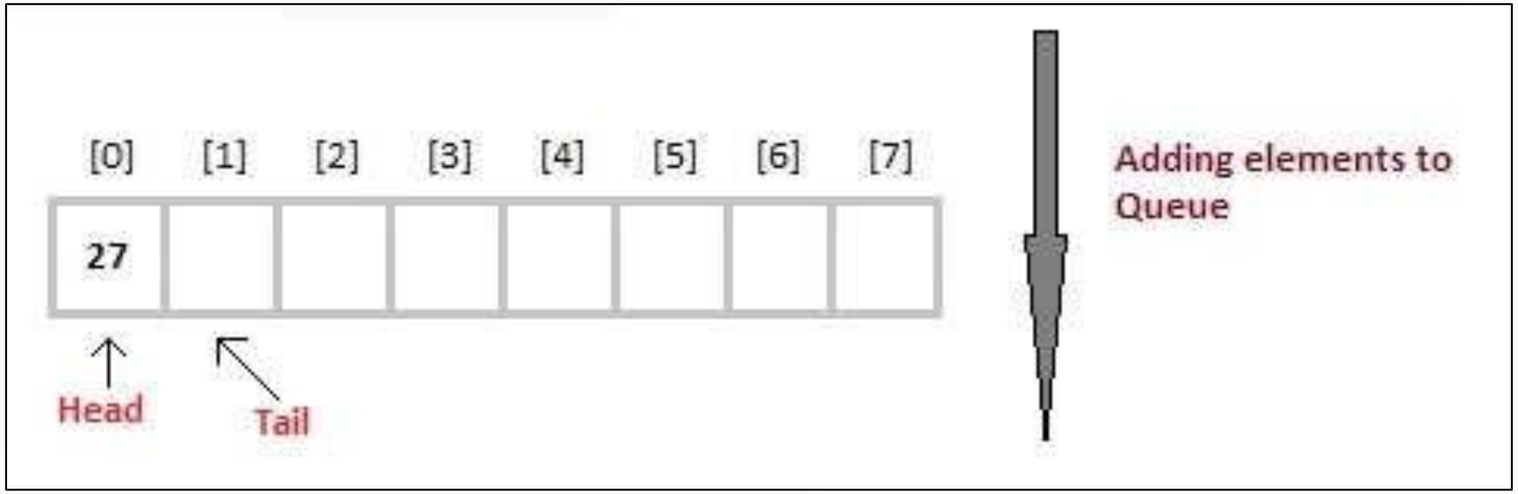
- Queue can be implemented using an Array, Stack or Linked List.
- The easiest way of implementing a queue is by using an **Array.**
- Initially the **head**(FRONT) and the **tail**(REAR) of the queue points at the first index of the array (*starting the index of array from 0*).



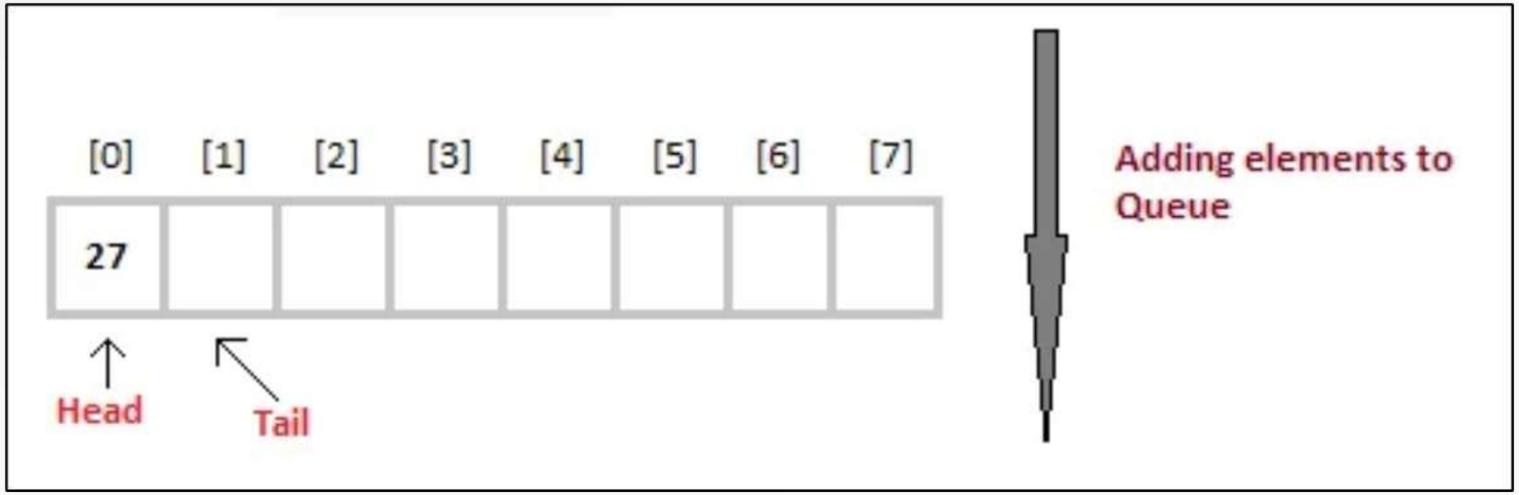
- As we add elements to the queue, the **tail** keeps on moving ahead, always pointing to the position where the next element

Implementation of Queue Data Structure



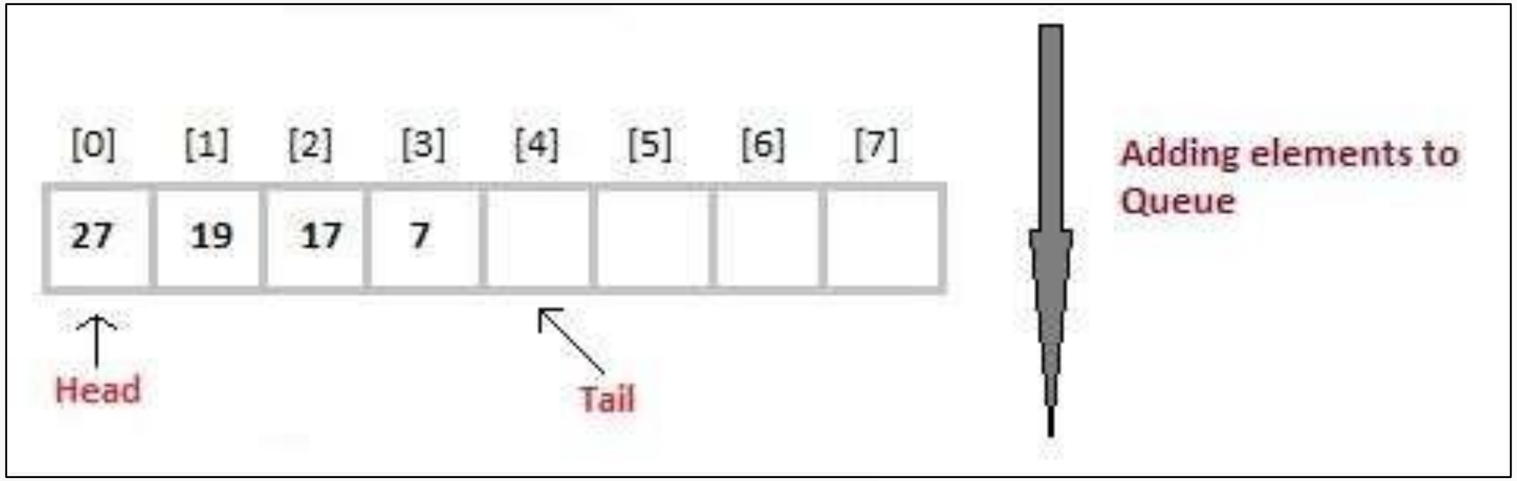


Implementation of Queue Data Structure



[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
27							

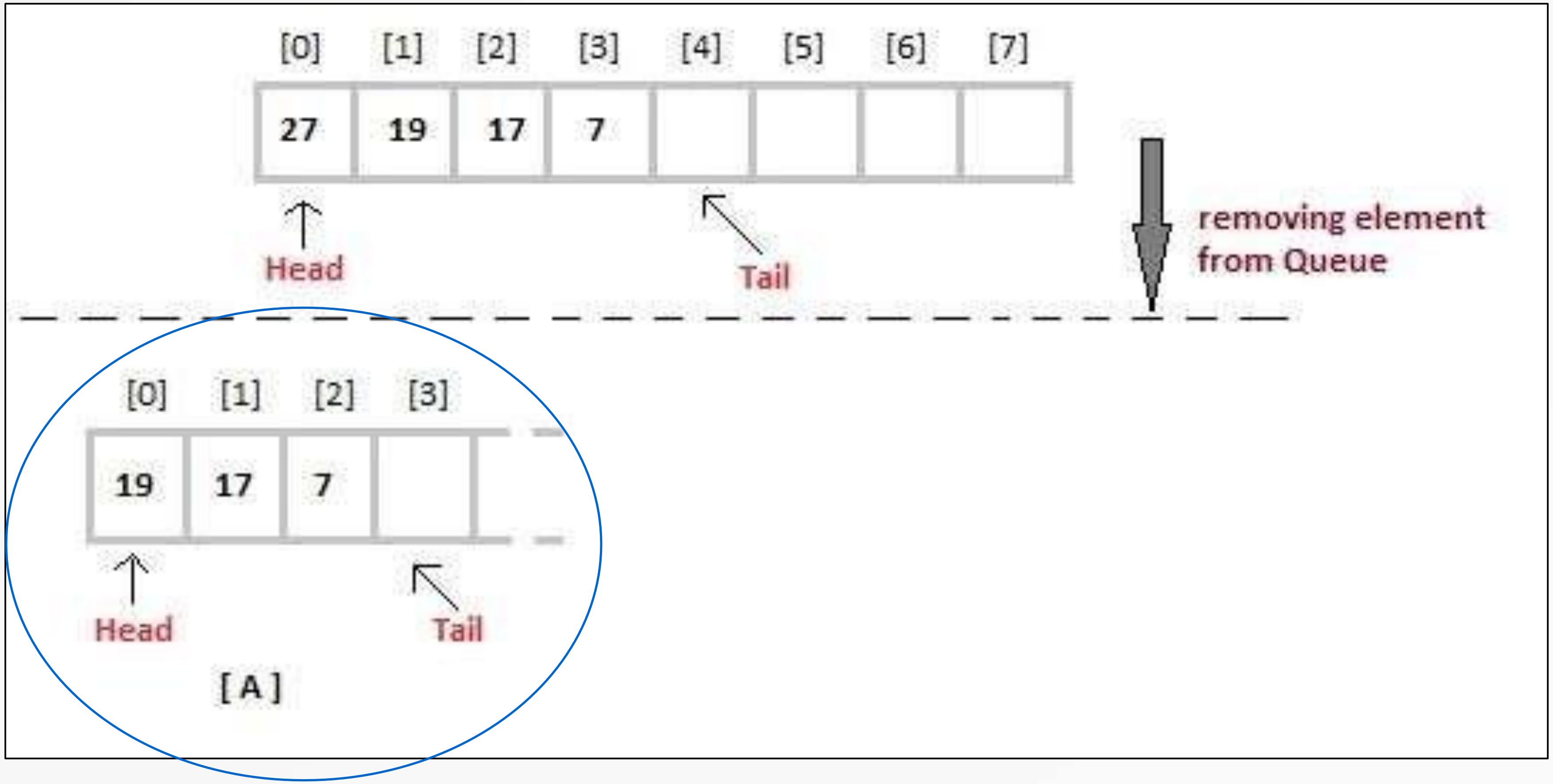
Adding elements to Queue



[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]
27	19	17	7				

Adding elements to Queue

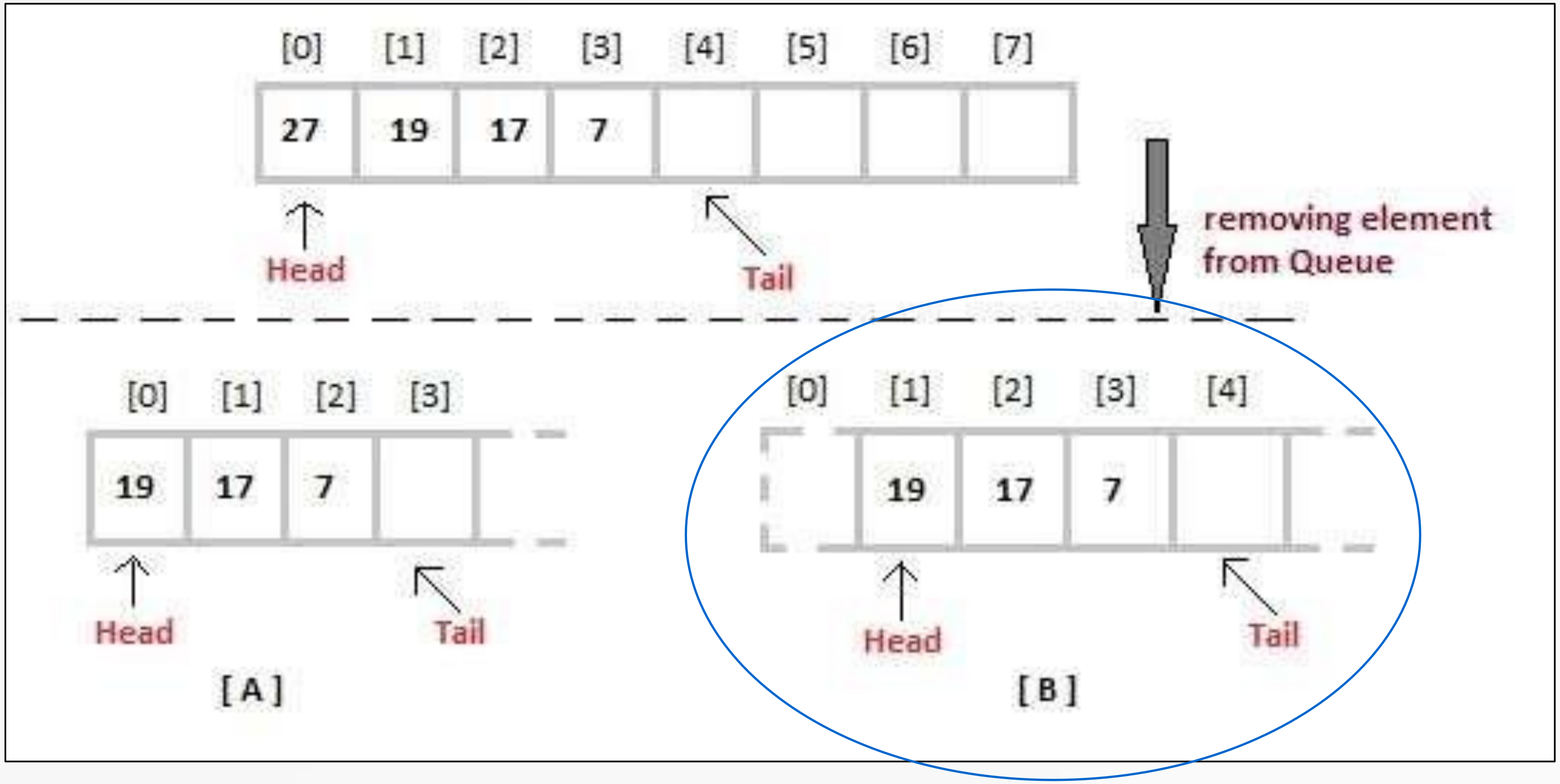
Implementation of Queue Data Structure



In [A] approach, we remove the element at head position, and then one by one shift all the other elements in forward position.

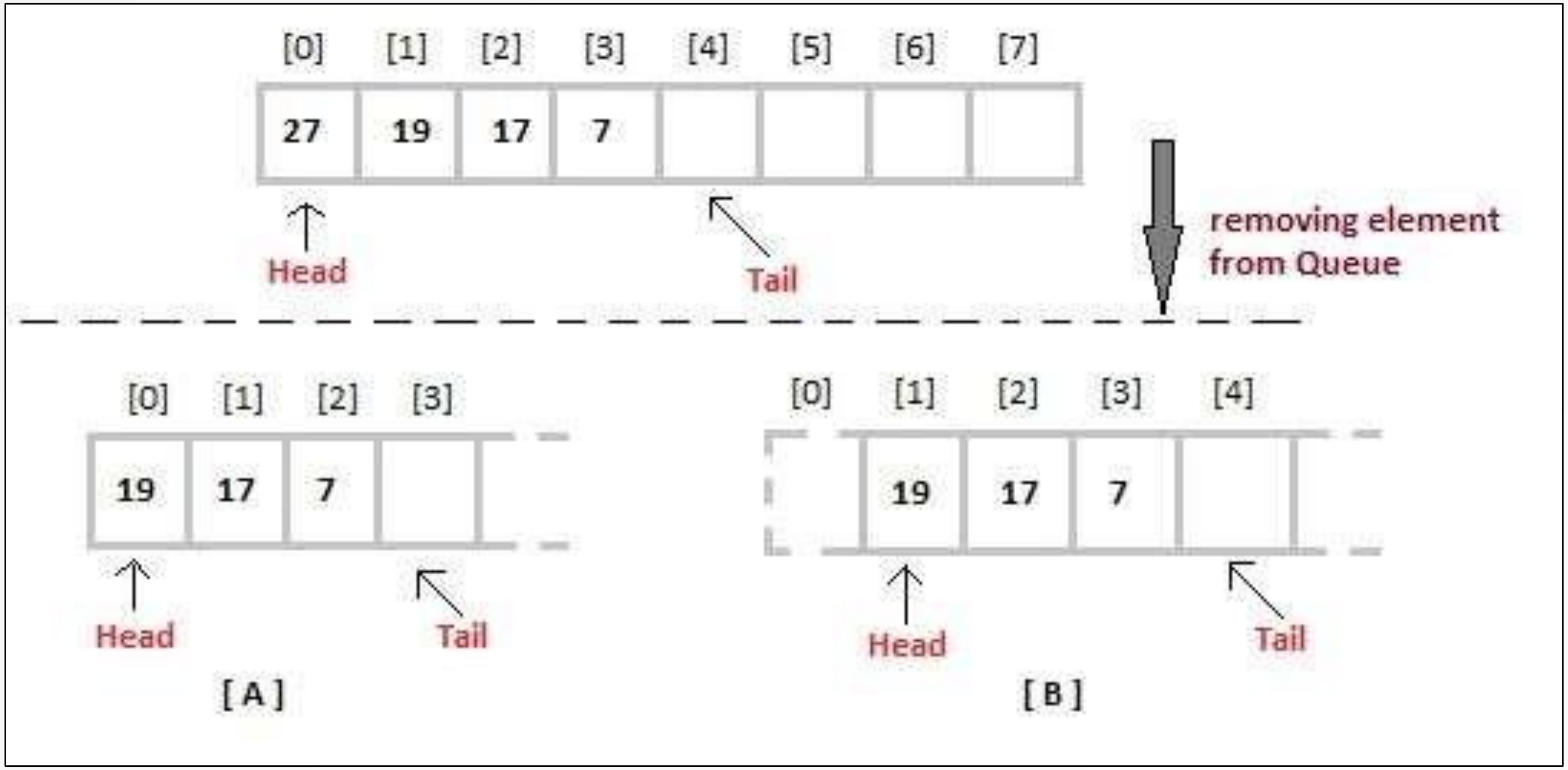
In approach [B] we remove the element from head position and then move head to the next position.

Implementation of Queue Data Structure



In approach [B] we remove the element from head position and then move head to the next position.

Implementation of Queue Data Structure



Basic Operation

- **enqueue()**
  - add (store) an item to the queue.
- **dequeue()**
  - remove (access) an item from the queue.
- **peek()**
  - Gets the element at the front of the queue without removing it.
- **isfull()**
  - Checks if the queue is full.
- **isempty()**
  - Checks if the queue is empty.

Sample Program

Sample program

```
#include <iostream>
#include<queue>

using namespace std;                              1

int main()
{
    queue<int> myQueue;
    int data;

    for(int  i =0; i<5; i++ ) {
        cout<<"Enqueuing "<<i<< " :";             2
        cin>>data;

        myQueue.push(data);
    }
    cout<< "Size of myQueue: "<< myQueue.size()<<endl;
    cout<< "The back of myQueue: "<< myQueue.back()<<endl;        3
    cout<< "The front of myQueue: "<< myQueue.front()<<endl;


    while(!myQueue.empty())
    {
        cout<<"Dequeuing "<<myQueue.front()<<endl;       4
        myQueue.pop();
    }


    return 0;
}
```

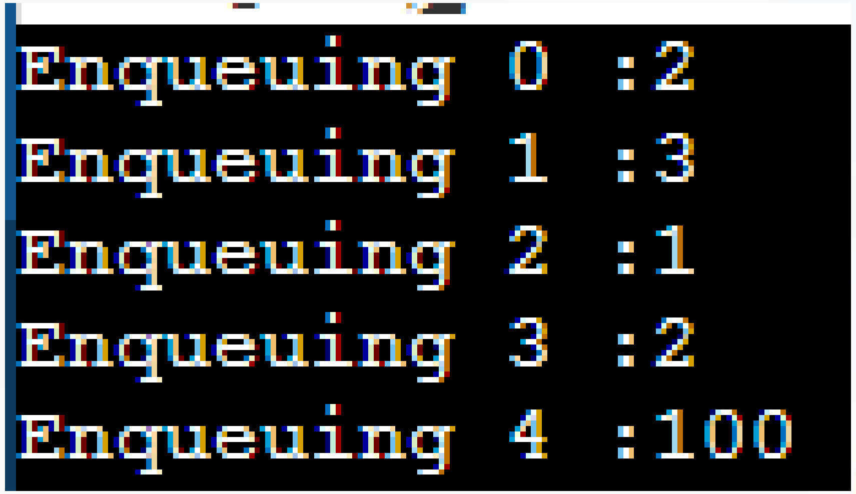
Sample program

```
#include <iostream>
#include<queue>

using namespace std;

int main()
{
    queue<int> myQueue;
    int data;
```

Sample program

```
#include <iostream>
#include<queue>

using namespace std;

int main()
{
    queue<int> myQueue;
    int data;
```

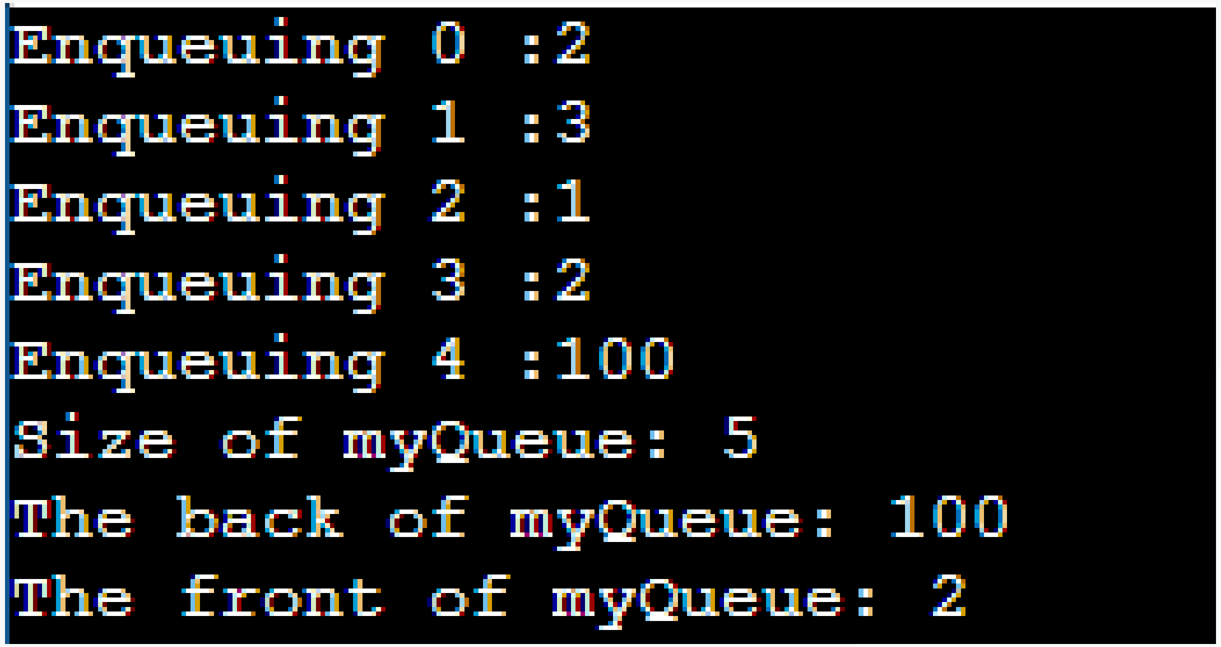
```
for(int  i =0; i<5; i++ ) {
        cout<<"Enqueuing "<<i<< " :";
        cin>>data;

        myQueue.push(data);
   }
```



```
Enqueuing 0 :2
Enqueuing 1 :3
Enqueuing 2 :1
Enqueuing 3 :2
Enqueuing 4 :100
```

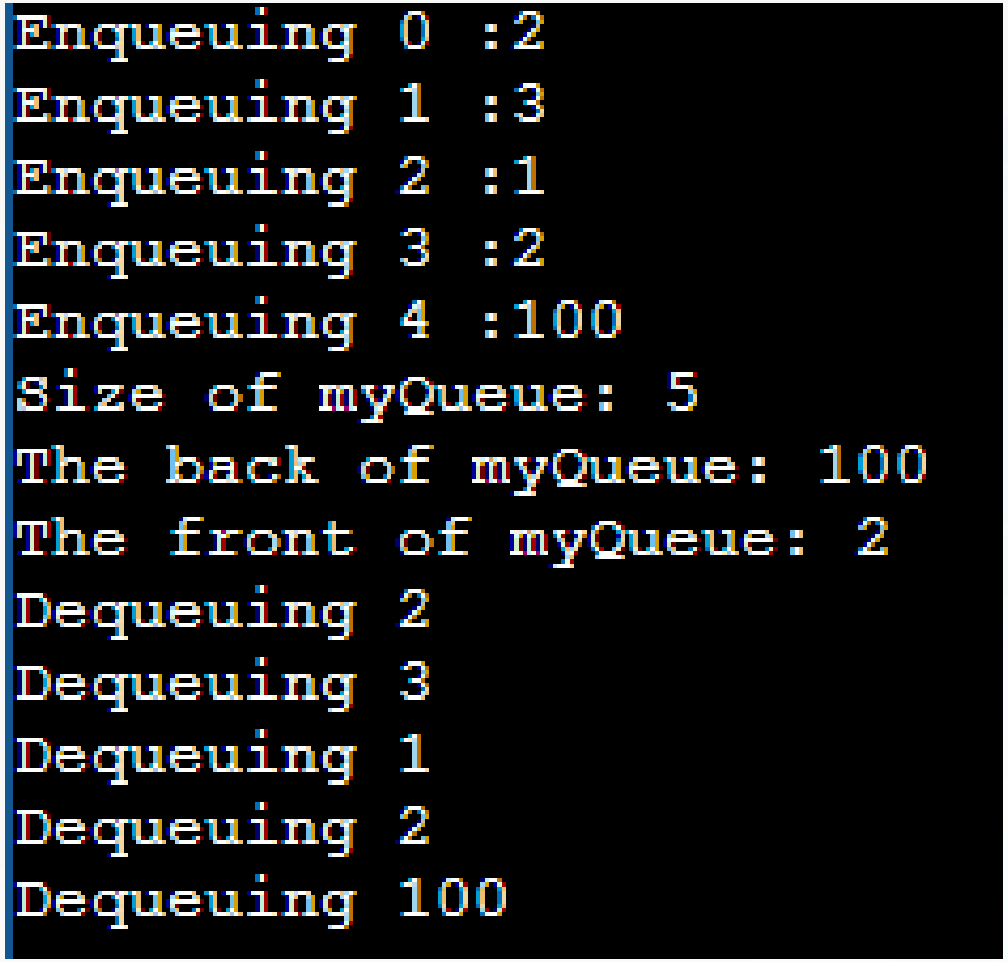
Sample program

```
for(int  i =0; i<5; i++ ) {
        cout<<"Enqueuing "<<i<< " :";
        cin>>data;
        myQueue.push(data);      }
cout<< "Size of myQueue: "<< myQueue.size()<<endl;
cout<< "The back of myQueue: "<< myQueue.back()<<endl;
cout<< "The front of myQueue: "<< myQueue.front()<<endl;
```



```
Enqueuing 0 :2
Enqueuing 1 :3
Enqueuing 2 :1
Enqueuing 3 :2
Enqueuing 4 :100
Size of myQueue: 5
The back of myQueue: 100
The front of myQueue: 2
```

```
while(!myQueue.empty())
    {
        cout<<"Dequeuing "<<myQueue.front()<<endl;
        myQueue.pop();
    }
```



```
Enqueuing 0 :2
Enqueuing 1 :3
Enqueuing 2 :1
Enqueuing 3 :2
Enqueuing 4 :100
Size of myQueue: 5
The back of myQueue: 100
The front of myQueue: 2
Dequeuing 2
Dequeuing 3
Dequeuing 1
Dequeuing 2
Dequeuing 100
```

References

- https://www.geeksforgeeks.org/queue-data-structure/
- https://www.tutorialspoint.com/data_structures_algorithms/dsa_queue.htm
- https://www.studytonight.com/data-structures/queue-data-structure

Thank you