# Static Queues

March 28, 2022

## 1 Static Queues

This document provides and explains a simulation of a static queue

The function `createQueue`'s name is fairly descriptive in terms of its function. It takes in a integer, which will be the queue's length, a queue containing instances of `None` is then created with the correct length. It then returns a tuple containing the queue list, front and rear pointers and the maximum queue length, the length of the queue list.

[1]:
```
def createQueue(queueLength: int) -> tuple:
    queue = [None for i in range(queueLength)]
    front = 0
    rear = -1
    queueMaxLength = len(queue)

    return queue, front, rear, queueMaxLength
```

Takes in `front` and `rear` and calculates the length of the queue, then compares this length to the maxminum length to the queue. If the size is greater than or equal to the queueMaxLength then it returns `True`, else it returns `False`

[2]:
```
def isFull(front: int, rear: int, queueMaxLength: int) -> bool:
    size = rear - front + 1
    if size >= queueMaxLength:
        return True

    return False
```

Takes in `front` and `rear` and calculates the length of the queue, then compares this length to the maxminum length to the queue. If the size is less than or equal to `0` then it returns `True`, else it returns `False`

[3]:
```
def isEmpty(front: int, rear: int) -> bool:
    size = rear - front
    if size < 0:
        return True

    return False
```

The enQueue function increments the value of the rear pointer by 1, then sets queue[rear] to the new item. It then returns the queue as well as the new rear value in a tuple.

```
[4]: def enQueue(queue: list, front: int, rear: int, queueMaxLength: int,␣
     ↪itemToEnQueue) -> tuple:
         if isFull(front, rear, queueMaxLength) is True:
             print(f'Failed to enqueue {itemToEnQueue} as queue ({", ".join(queue)})␣
     ↪is full.')
             return

         rear += 1
         queue[rear] = itemToEnQueue

         return queue, rear
```

The function deQueue stores the element at the position indicated by the front pointer's value. This position is then set to None and the deQueuedItem, queue and front are returned in a tuple.

```
[5]: def deQueue(queue: list, front: int, rear: int, queueMaxLength: int) -> tuple:
         if isEmpty(front, rear) is True:
             print(f'Failed to dequeue from queue, as queue is empty.')
             return

         deQueuedItem = queue[front]
         queue[front] = None

         front += 1

         return deQueuedItem, queue, front
```

## 1.1 Testing

### 1.1.1 Testing Functions

```
[6]: def logQueue(queue: list, front: int, rear: int, queueMaxLength: int):
         print(f'Queue: {queue}')
         print(f'isEmpty: {isEmpty(front, rear)}')
         print(f'isFull: {isFull(front, rear, queueMaxLength)}')
```

Initialise the variables queue, front, rear and queueMaxLength, unpacking the tuple returned by the createQueue function and setting their values appropriately.

### 1.1.2 Queues Testing

```
[7]: queue, front, rear, queueMaxLength = createQueue(4)
```

```
[8]: print(queue, front, rear, queueMaxLength)
```

```
[None, None, None, None] 0 -1 4
```

Adding `'Moss'`, `'Roy'`, `'Jen'` and `'Douglas'` to the queue.

```
[9]: queue, rear = enQueue(queue, front, rear, queueMaxLength, 'Moss')
     queue, rear = enQueue(queue, front, rear, queueMaxLength, 'Roy')
     queue, rear = enQueue(queue, front, rear, queueMaxLength, 'Jen')
     queue, rear = enQueue(queue, front, rear, queueMaxLength, 'Douglas')
```

```
[10]: logQueue(queue, front, rear, queueMaxLength)
```

```
Queue: ['Moss', 'Roy', 'Jen', 'Douglas']
isEmpty: False
isFull: True
```

```
[11]: deQueuedItem, queue, front = deQueue(queue, front, rear, queueMaxLength)
      logQueue(queue, front, rear, queueMaxLength)
```

```
Queue: [None, 'Roy', 'Jen', 'Douglas']
isEmpty: False
isFull: False
```

```
[12]: deQueuedItem, queue, front = deQueue(queue, front, rear, queueMaxLength)
      deQueuedItem, queue, front = deQueue(queue, front, rear, queueMaxLength)
      deQueuedItem, queue, front = deQueue(queue, front, rear, queueMaxLength)
```

```
[13]: logQueue(queue, front, rear, queueMaxLength)
```

```
Queue: [None, None, None, None]
isEmpty: True
isFull: False
```