# Idea

The Idea behind the ADT of a Queue is to provide a collection of data with two primary operations:

- *push*: Push a new item on top of the Queue.
- *pop*: Take the item on top of the Queue off.

# Definition

We define the ADT as the following 5-Tuple:

$$\mathcal{D} = (N, P, Fs, Ts, Ax),$$

where the components are defined as follows:

1. $N :=$ Queue
2. $P := \{\text{Element}\}$
3. $Fs := \{\text{queue, enqueue, dequeue, peek, length}\}$
4. $Ts$ is the set containing the following type specifications:
   (a) queue : Queue
   (b) length : Queue $\to \mathbb{N}_0$
   (c) enqueue : Queue $\times$ Element $\to$ Queue
   (d) dequeue : Queue $\to$ Queue $\cup \{\Omega\}$
   (e) peek : Queue $\to$ Element $\cup \{\Omega\}$
5. $Ax$ is the set containing the following axioms.
   $\forall Q \in$ Queue : $x, y \in$ Element :
   (a) queue().peek() $= \Omega$
   (b) queue().length() $= 0$
   (c) $Q$.enqueue($x$).length() $=$ $Q$.length() + 1
   (d) $Q$.length() $> 0 \to Q$.dequeue().length() $=$ $Q$.length() - 1
   (e) $Q$.length() $> 0 \to Q$.enqueue($x$).dequeue() $=$ $Q$.dequeue().enqueue($x$)
   (f) queue().enqueue($x$).dequeue() $=$ queue()
   (g) queue().dequeue() $= \Omega$
   (h) $Q$.length() $> 0 \to Q$.enqueue($x$).peek() $=$ $Q$.peek()
   (i) $Q$.length() $= 0 \to Q$.enqueue($x$).peek() $= x$

# Implementation

TBD