

# Lecture 07 Exercise Solutions

Mark Eramian

## Exercise 1

---

**Name:** Queue<G>

---

**Sets:**

$Q$  : set of queues containing items from  $G$

$G$  : set of items that can be in the queue

$B$  : {true, false}

$\mathbb{N}_0$ : set of non-negative integers

---

**Signatures:**

newQueue<G>(n) :  $\mathbb{N}_0 \rightarrow Q$

$Q$ .isEmpty:  $\rightarrow B$

$Q$ .isFull:  $\rightarrow B$

$Q$ .add(g):  $G \rightarrow Q$

$Q$ .remove:  $\rightarrow G$

---

**Preconditions:** For all  $q \in Q, g \in G$

newQueue<G>(n):  $n > 0$

$q$ .isEmpty: none

$q$ .isFull: none

$q$ .add(g):  $q$  is not full

$q$ .remove:  $q$  is not empty

---

**Semantics:** For  $q \in Q, g \in G, n \in \mathbb{N}_0$

newQueue<G>(n) : create a queue of items from  $G$  with capacity  $n$

$q$ .isEmpty: returns true if  $q$  is empty, false otherwise

$q$ .isFull: return true if  $q$  is full, false otherwise

$q$ .add(g): enqueues  $g$  at the back of the queue

$q$ .remove: removes then returns the item at the front of the queue

## Exercise 2

---

**Name:** Stack<G>

---

**Sets:**

$S$  : set of stacks containing items from  $G$

$G$  : set of items that can be in the stack

$B$  : {true, false}

$\mathbb{N}_0$ : set of non-negative integers

---

**Signatures:**

newStack<G>(n) :  $\mathbb{N}_0 \nrightarrow S$

S.isEmpty:  $\rightarrow B$

S.isFull:  $\rightarrow B$

S.push(g):  $G \nrightarrow S$

S.pop:  $\nrightarrow S$

S.top:  $\nrightarrow G$

---

**Preconditions:** For all  $s \in S, g \in G, n \in \mathbb{N}_0$

newStack<G>(n):  $n > 0$

s.isEmpty: none

s.isFull: none

s.push(g):  $s$  is not full

s.pop:  $s$  is not empty

s.top:  $s$  is not empty

---

**Semantics:** For  $s \in S, g \in G, n \in \mathbb{N}_0$

newStack<G>(n) : create an empty stack of items from  $G$  with capacity  $n$

s.isEmpty: returns true if  $s$  is empty, false otherwise

s.isFull: return true if  $s$  is full, false otherwise

s.push(g): adds  $g$  to the top of the stack

s.pop: removes the element at the top of the stack

s.top: return the element at the top of the stack