

```

void appendV1 (QueueOfT& r, QueueOfT& g) // Using r for receiver, g for giver
//! updates r
//! clears g
//! requires: true
//! ensures: r = #r * #g

```

S	Code	Assume		Confirm
0		true		$r_0 * g_0 = r_0 * g_0$ ①
	while(g.length() > 0) { //! updates g, r //! maintains //! $r * g = \#r * \#g$ //! decreases g			
1		$ g_1 > 0 \wedge$ $r_1 * g_1 = r_0 * g_0$ ③		
	T y;			
2		T.Init(y2)	Unchanged r, g	$g_2 \neq \langle \rangle$
	g.dequeue(y);			
3		$g_3 = g_2[1, g_2) \wedge$ $\langle y_3 \rangle = g_2[0, 1)$	Unchanged r	
	r.enqueue(y);			
4		T.Init(y4) ^ $r_4 = r_3 * \langle y_3 \rangle$	Unchanged g	$ g_4 < g_1 \wedge$ $r_4 * g_4 = r_0 * g_0$ ②
	}			
5		$\sim(g_5 > 0) \wedge$ $r_5 * g_5 = r_0 * g_0$ ④		$r_5 = r_0 * g_0 \wedge$ $g_5 = \langle \rangle$
7				

Loop invariant related reasoning:

- Must confirm it holds at ① and ②
- Get to assume it holds at ③ and ④
- Allows us to reason about the loop as if it were a single statement

