Loop Tracing Table for *injectAtFront*

Name: _____

To do:

1. Fill in the loop tracing table found on the next page for the operation *injectAtFront*
   1.1. *injectAtFront*'s full implementation is found below
   1.2. The incoming data specified in the tracing table will require 3 passes of the loop body
   1.3. Reference instructional materials found in Week #6 of the CSSE373 Moodle site:
      01B - Synthesizing a Loop Invariant Using a Tracing Table (TT)
2. Fill in the *while* loop's internal contract, i.e., the *updates*, *maintains*, and *decreases* clauses

```
template <class T, class QueueOfT>
void InjectCapability1<T, QueueOfT>::inject(T& x)
//! updates self
//! clears x
//! ensures: self = <#x> * #self
{

    QueueOfT t;

    t.enqueue(x)
    while (self.length() > 0) {
        //! updates   _____________________________
        //! maintains _____________________________
        //! decreases _____________________________
        T y;

        self.dequeue(y);
        t.enqueue(y);
    } // end while
    self.transferFrom(t);
}   // inject
```

Tracing Table	**Original incoming values to the loop:** *#self = <1,5,7> #t = <47>* *self = ??????? t = ????*		
`while (self.length() > 0) {`	**1st pass column**	**2nd pass column**	**3rd pass column**
State 0	*self0:1 = <1,5,7>* *t0:1 = <47>* *y0:1 = 0* `// assume: Integer y;`	*self0:2 =* *t0:2 =* *y0:2 =*	*self0:3 =* *t0:3 =* *y0:3 =*
`self.dequeue(y);`			
`t.enqueue(y);`			
State 1	*self1:1 =* *t1:1 =* *y1:1 =*	*self1:2 =* *t1:2 =* *y1:2 =*	*self1:3 =* *t1:3 =* *y1:3 =*
`} // end while`			