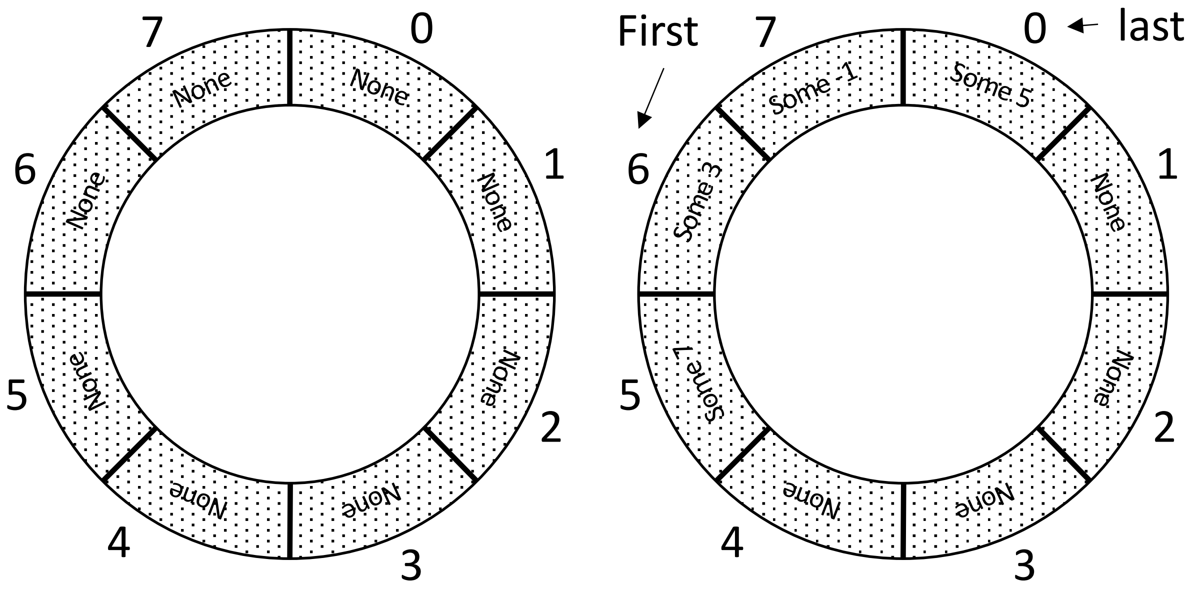
Learning to Program with F#
Exercises

Department of Computer Science
University of Copenhagen

Jon Sporring, Martin Elsman, Torben Mogensen, Christina Lioma

November 24, 2022



(a) (b)

Figure 1: A cyclic queue of length 8. Integers on the outside of the figure are position indices. (a) The initial state with no data and where the first and last are also set to `None`. (b) A queue after enqueueing the sequence $7, 3, -1, 5$ and then dequeuing once such that the first element is 3 and the last is 5. The next element to be added will be in position 1.

0.1 cyclicQueue

0.1.1 Teacher's guide

Emne rekursion, grafik og winforms

Sværhedsgrad Middel

0.1.2 Introduction

In this assignment, you are to work with cyclic queues. A cyclic queue is a queue with fixed storage space allocated for it, such as an array of constant length, and two points pointing to the first and last element in the queue. As an example, consider a cyclic queue of length 8 as illustrated in Figure 1(a). When the queue maintains the variables `first` and `last`, where `first` points to the position of the element in the front of the queue, and `last` points to the position of the element in the back of the queue, which also is the last element added to the queue. Initially, point pointers are `None`. If the queue is not full, then when enqueueing values to the queue, `last` is cyclicly incremented by 1, and the value is stored in that position. If the queue is not empty, then dequeuing values from the queue, the value at position `first` is returned and lstinlinefirst is cyclicly incremented by 1.

In this assignment, you are to make a module that implements the abstract datatype known as a *cyclicQueue* for integers using imperative programming and which has the following interface `cyclicQueue.fsi`:

```
module cyclicQueue
```

```
type Value = int

/// <summary>Create or clear the cyclic queue</summary>
/// <param name="n">The maximum number of elements</param>
val create: n: int -> unit

/// <summary>Add an element to the end of a queue</summary>
/// <param name="e">an element</param>
/// <returns>True if the queue had space for the element</returns>
val enqueue: e: Value -> bool

/// <summary>Remove the element in the front position of the
   queue</summary>
/// <returns>The first element in q or None if the queue is
   empty</returns>
val dequeue: unit -> Value option

/// <summary>Check if the queue is empty</summary>
/// <returns>True if the queue is empty</returns>
val isEmpty: unit -> bool

/// <summary>Get the length of the queue</summary>
/// <returns>The number of elements in the queue</returns>
val length: unit -> int

/// <summary>The queue on string form</summary>
/// <returns>A string representing the queue's elements</returns>
val toString: unit -> string
```

0.1.3 Exercise(s)

0.1.3.1: (a) make an implementation of `cyclicQueue.fsi` called `cyclicQueue.fs`. The implementation must use mutable value for `first` and `last`, and a mutable array `q: Value option[]`

(b) Write an application that tests each function. Consider whether you can make your functions cast exceptions.

(c) In comparison with a purely functional implementation of a general queue, what are the advantages and disadvantages of this implementation?