Queue

The queue is a FIFO (First-in First-out) data structure

Elements are added at the front of the queue and removed from the rear

The only data element that can be removed is the least recently added element

Queue ADT

Specification

Elements: Queue elements can be of any type, but we will assume QueueElement

Structure: Any mechanism for determining the elements order of arrival into the queue

Queue ADT Continued

Domain: The number of queue elements is bounded. A queue is considered full if the upper-bound is reached. A queue with no elements is considered empty.

type Queue;

Operations: There are six operations as follows:

Queue ADT Continued

function create (q: Queue, isCreated: boolean)
results: if q cannot be created, isCreated is false; otherwise, isCreated is true,the queue is created and is empty

function terminate (q: Queue)
results: queue q no longer exists

Queue ADT Continued

function isFull (q: Queue)
results: returns true if the queue is full; otherwise false is returned

function isEmpty (q: Queue)
results: returns true if the queue is empty; otherwise, false is returned

function enqueue (q: Queue, e: QueueElement)
requires: isFull (q) is false
results: element e is added to the front of the queue as the most recently added element

Queue ADT Continued

function dequeue (q: Queue, e: QueueElement)
requires: isEmpty(q) is not false
results: The least recently added element is removed and assigned to e

Queue Implementation

Problem: Write queue.h.

Problem: After we agree on queue.h, write create, terminate, isFull, isEmpty, enqueue, and dequeue.

Problem: Can you think of an application that requires a queue?