COMP 211: Lab 6
Fall, 2014

1 Survey

Please fill out the following mid-semester survey!

`https://www.surveymonkey.com/s/comp211`

2 Queues

To get the code for this task, go to the Assignments page and save the files and `lab06-queue.c0` and `lab06-queue-client.c0` to your COMP211 directory. You can load them by typing

```
% coin -d lab06-queue.c0 lab06-queue-client.c0
```

Queues are a data structure similar to stacks, except they have different behavior: for stacks, the last element added to the stack is the first to be popped off, whereas for queues the *first* person to get in line is the first person to be dequeued.

For this lab, we consider four operations on queues:

- check whether a queue is empty
- make an empty queue
- enqueue an element
- dequeue an element (returns the oldest element in the queue)

This wishlist translates into the following interface:

```
bool queue_empty(queue Q);      /* O(1) */

queue queue_new()               /* O(1) */
/*@ensures queue_empty(\result); @*/;

void enq(queue Q, string s)     /* O(1) */
/*@ensures !queue_empty(Q);@*/;

string deq(queue Q)               /* O(1) */
/*@requires !queue_empty(Q);@*/;
```

Your task is to implement this interface.

Task 2.1 Define a struct named `queue_header` representing a queue.

We suggest represent a queue by a struct with three things:

- an array `data`. This can have a fixed capacity of 1000 for this lab.
- an int `front`, which represents the position that the next element will be dequeued from (i.e. it represents the "first person in line")
- an int `back`, which represents the position that the next enqueued element will be put in (i.e. the "next position in line")

Task 2.2 A `queue_header` is "good" if the length of `data` is 1000, and `front` and `back` are in-bounds for `data`.

Define a data structure invariant function

```
bool is_queue(queue Q)
```

that captures these invariants.

Task 2.3 Implement the four functions in the queue interface, `queue_empty`, `queue_new`, `enq`, and `deq`. Each function should require and ensure the data structure invariants. Your code should call `error("out of space in the queue")` when the capacity is exceeded (like we did for stacks).

We have provided a function `test_queue` in `queue-client.c0` for testing.

Bonus Tasks

Task 2.4 For the code you wrote above, there is probably a relatively simple way to make `enq` run out of space less often. Implement this. How does it change the running time of `enq`?

Task 2.5 Extend your implementation so that `enq` never runs out of space. How does this change the running time of `enq`?