CS 180 - Queues

Note Title 9/13/2010

Announcements

- HW due Monday by start of class
- Next assignment will be out Friday (program on stacks)

Queue

list - stores in FIFO
(like a line)

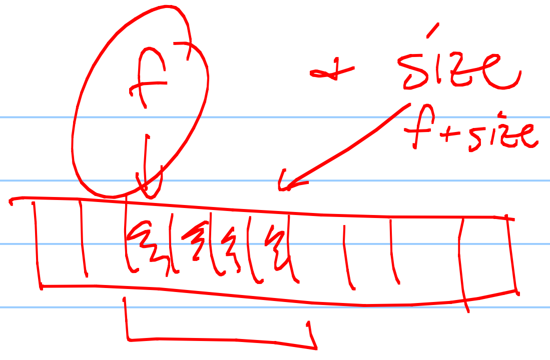
(stacks were LIFO)

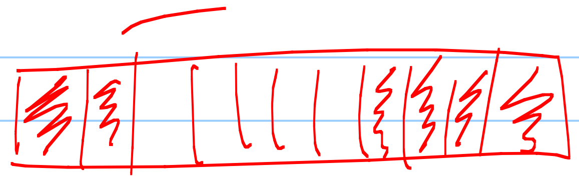
Alright - let's think about the setup:

```
template <typename Object>
 Class  Queue  {
     public :

         int size() const;

         bool isEmpty() const;

         const Object& front() const;

         void enqueue(Object obj);

         Object dequeue();
 };
```

How to implement?

use an array





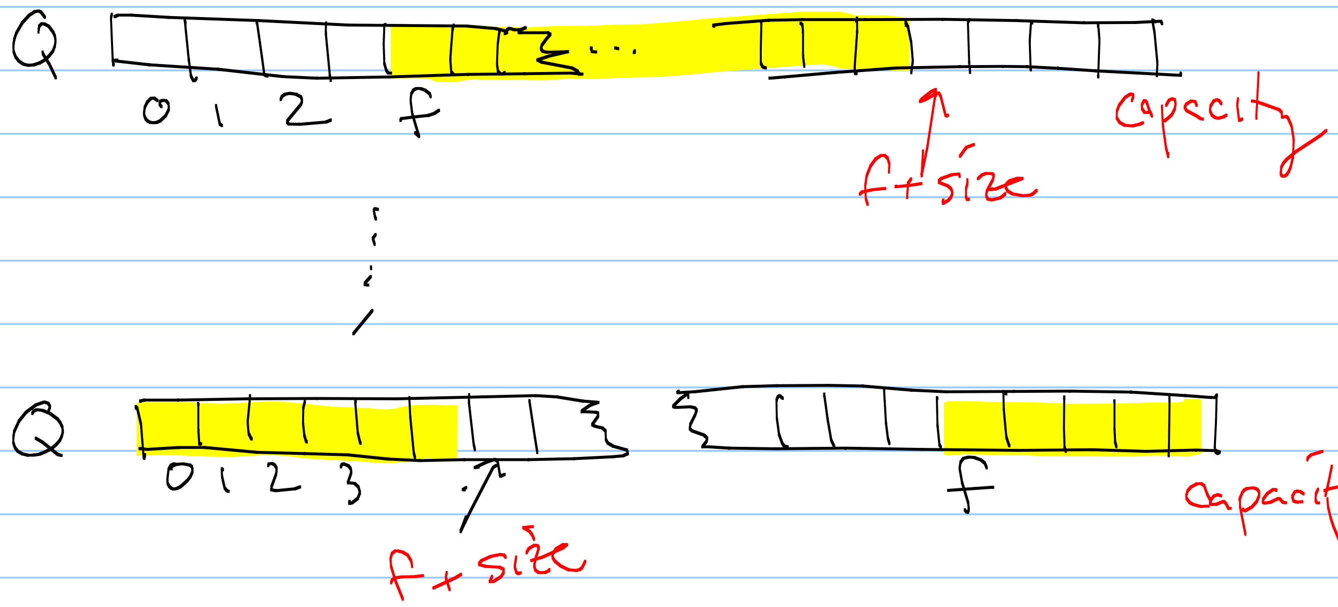
Private data:

```
Object * Q; <-
int    f;
int    size;
int    capacity;
```

in constructor

Q = new Object[capacity];

Wrapping Around:



Two options:

✗ - A lot of if statements

- Modular arithmetic: remainders

$1 \bmod 3 = 1$

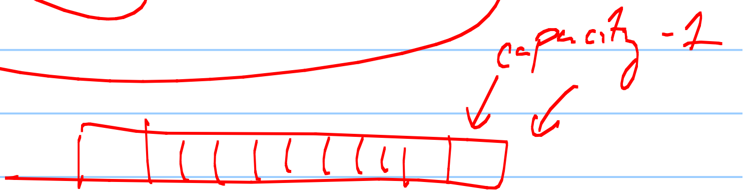
$4 \bmod 3 = 1$

$5 \bmod 3 = 2$

$11 \bmod 3 = 2$

$\%$

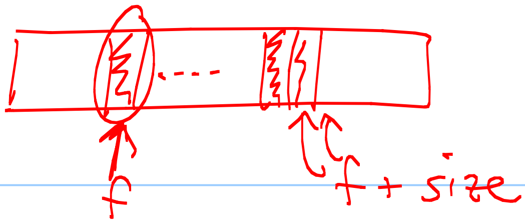
$$\begin{array}{r} 0 \\ 3\overline{)1} \\ \underline{0} \\ \text{r. } 1 \end{array} \qquad \begin{array}{r} 1 \\ 3\overline{)4} \\ \underline{3} \\ 1 \end{array}$$



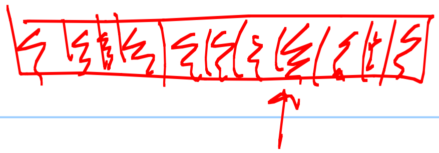
mod capacity

Pseudocode

```
isEmpty():
    return (size == 0);

size():
    return size;
```

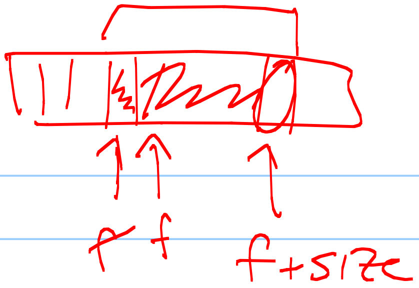


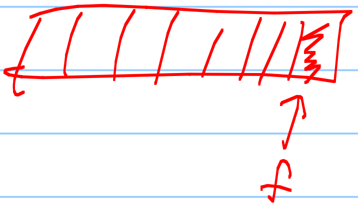
```
enqueue (element): {
  if(((f+ size)% capacity) == f)
       throw error
  else
     Q [(f + size) % capacity] = element;
     size ++;
}
```



f = f + ~~size~~ size % cap
size = 10

```
Object dequeue () {
    if (isEmpty())
        throw error;
    else {
        int oldf = f;

        f = (f+1) % capacity;
    →   size --;
        return Q[oldf];
    }
}
```



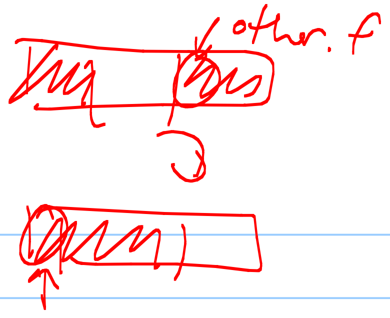


Actual code

(on webpage or in text)

Housekeeping Functions

- destructor
- operator =
- copy constructor



private:

```
void CopyData(const ArrayQueue& other) {
    // assuming this + other have same capacity
    f = 0;
    int walk = other.f;
    for (int i = 0; i < size; i++) {
        Q[i] = other.Q[walk];
        walk = (walk + 1) % capacity;
    }
}
```

```
                                        stack1 = stack2;
                                          ↑

ArrayQueue&  operator=(const ArrayQueue& other) {
   if (this != &other) {
       size = other.size;
       capacity = other.capacity;
       delete [] Q;
       Q = new Object[capacity];
       copyData(other);

   }
   return *this;
}
```

```
ArrayQueue  stack1(stack2);

ArrayQueue (const ArrayQueue & other) {
    size = other.size;
    capacity = other.capacity;
    Q = new Object[capacity];
    f = 0;
    copyData(other);
}

~ArrayQueue() {
    delete [] Q;
}
```