

CS 180 - Lecture 10

Announcements:

- Program 1 due Friday
- HW3 - up later today, due Wed.
- Review session on Monday in class
 - Practice exams will be handed out in class on Thurs./Fri., so be here one of those days
- Midterm - next Thurs (Lab next Wed.)

HW2 Solutions

(coded on other computer -
email me if you need a copy)

Extra credit:

$$X^n = X^{\lceil n/2 \rceil} \cdot X^{\lfloor n/2 \rfloor}$$

$$X^5 = X^2 \cdot X^3$$

$$X^{101} = X^{50} \cdot X^{51}$$

$$X^{100} = X^{50} \cdot X^{50}$$

$$2^{50}$$

$$2^2$$

$$2^4 = 2^2 \cdot 2^2$$

$$2^8 = 2^4 \cdot 2^4$$

⋮

$$2$$

3 mult.

Queues: Another way of storing a list
First in, first out (FIFO)

Two main functions:

enqueue(o): Insert object o at the rear of the queue
Input: object o, Output: None

dequeue(): Remove & return the object at the front of the queue
Input: None, Output: object o

Other operations

- size()


- isEmpty()

- first() (returns but doesn't remove the first element)

Operation	Output	Queue
enqueue(5)	-	<5>
enqueue(3)	-	<5, 3>
dequeue()	5	<3>
enqueue(7)	-	<3, 7>
dequeue()	3	<7>
isFront()	7	<7>
dequeue()	7	<>
dequeue()	error	<>
is Empty()	true	<>
enqueue(9)	-	<9>
enqueue(7)	-	<9, 7>
size()	2	<9, 7>
enqueue(3)	-	<9, 7, 3>
enqueue(5)	-	<9, 7, 3, 5>
dequeue()	9	<7, 3, 5>

Alright - let's think about the setup:

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

```
    int size() const;  function is an accessor
```

```
    bool isEmpty() const;
```

```
    const Object& front() const;
```

```
    void enqueue(Object obj);
```

```
    Object dequeue();
```

```
};
```

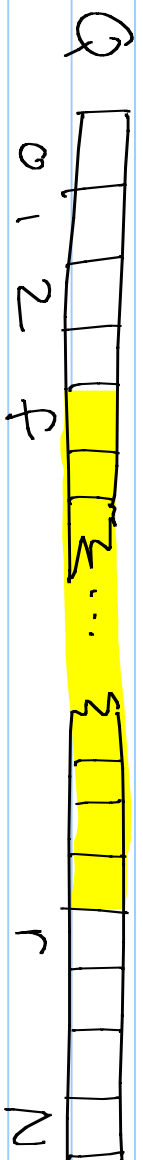
How should we implement?

arrays

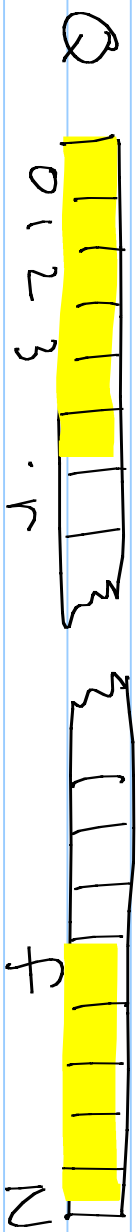
What issues might happen?

- maximum possible size

Wrap around:



⋮



Two options:

- A lot of IF statements

- Modular arithmetic: remainders

$$3 \bmod 3 = 0$$

$$1 \bmod 3 = 1$$

$$\rightarrow 4 \bmod 3 = 1$$

$$5 \bmod 3 = 2$$

$$N+1 \bmod N = 1$$

$N =$ maximum capacity, + I enqueue an element,

$$Q[r] = a_j;$$

$$r = (r+1) \% N;$$