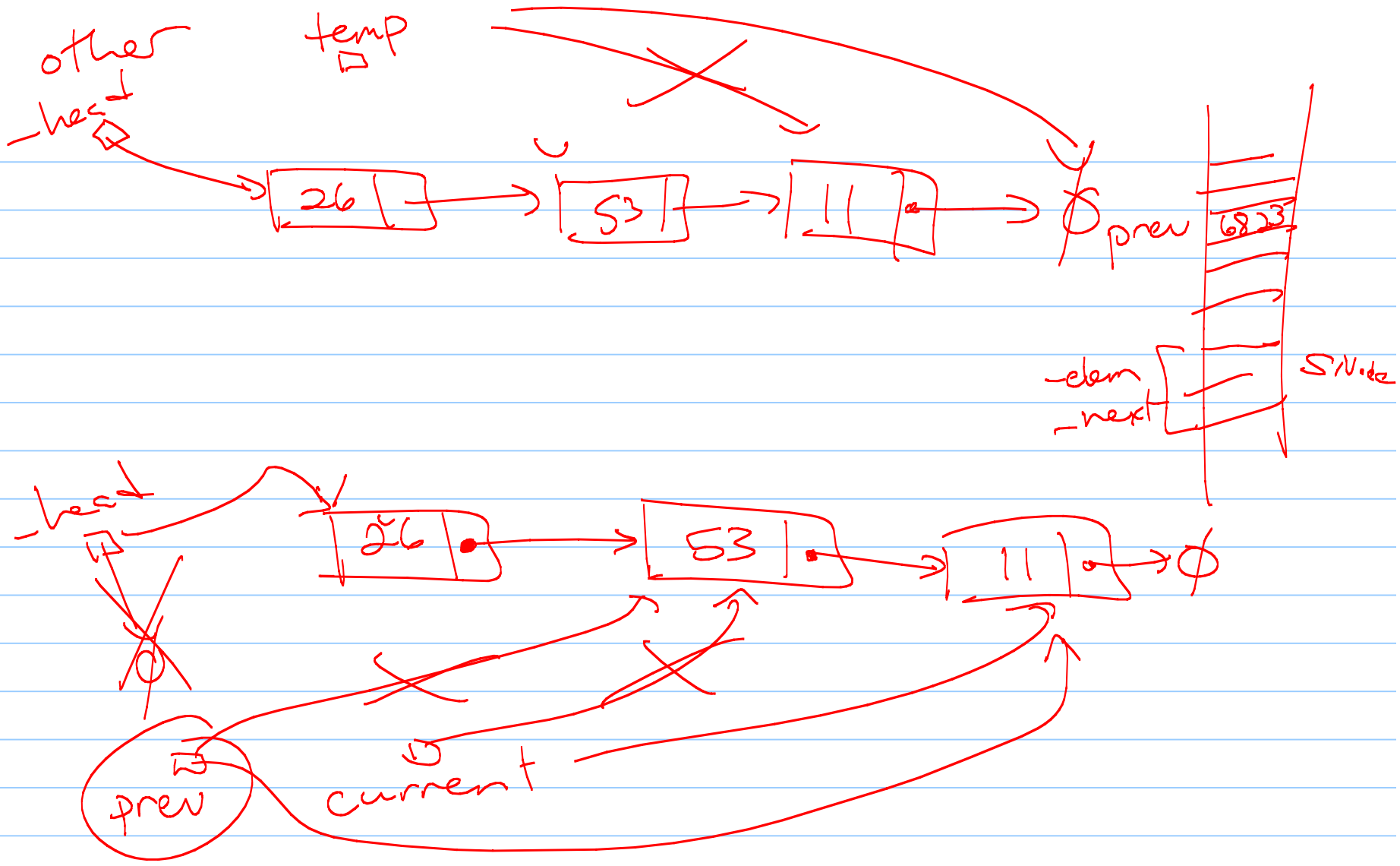
CS180 - Queues

Note Title 9/13/2010

Announcements

- Program due Sat.
- Next HW up tomorrow, due next Friday
- Midterm 1 will be the following week (Feb. 28 - March 4)
- Office hours: after class today and tomorrow 2-3pm

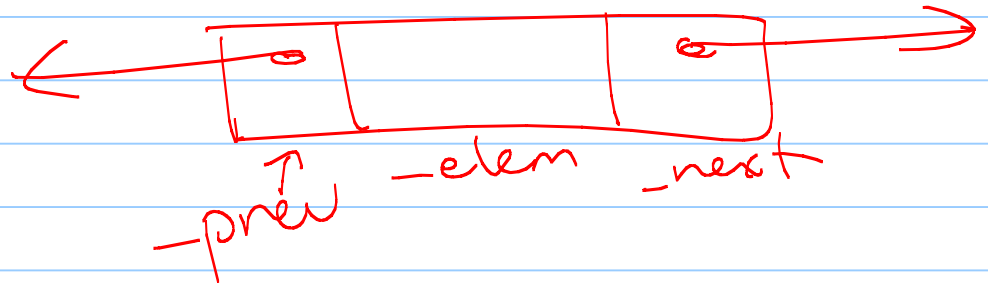
(Let me know if you plan to come!)


other
_head
temp
26
53
11
prev
6823
_elem
_next
SNode
_head
26
53
11
prev
current

Last time: ~~Operator=~~ Copy Constructor for SLinkedList

```
SLinkedList(const SLinkedList& other) {
    SNode<Object> * current, prev, temp;
    temp = other._head;
    _head = NULL;
→   while ( temp -> _next != NULL) {
        current = new SNode<Object>;
        current->_elem = temp -> _elem;
        if(_head == NULL)
            _head = current;
        else
            prev -> _next = current;
        temp = temp -> _next;
        prev = current;
    }
    prev -> _next = NULL;
}
```

Doubly Linked Lists



Queue

A line

First in, first out

(A list, with specific functionality)

(Queue.h)

Alright - let's think about the setup:

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

    int size() const;
    bool isEmpty() const;
    const Object& top() const;
    void push(Object obj);
    Object pop();

};
```

STL

(book calls these)

front

enqueue

dequeue

How to implement?

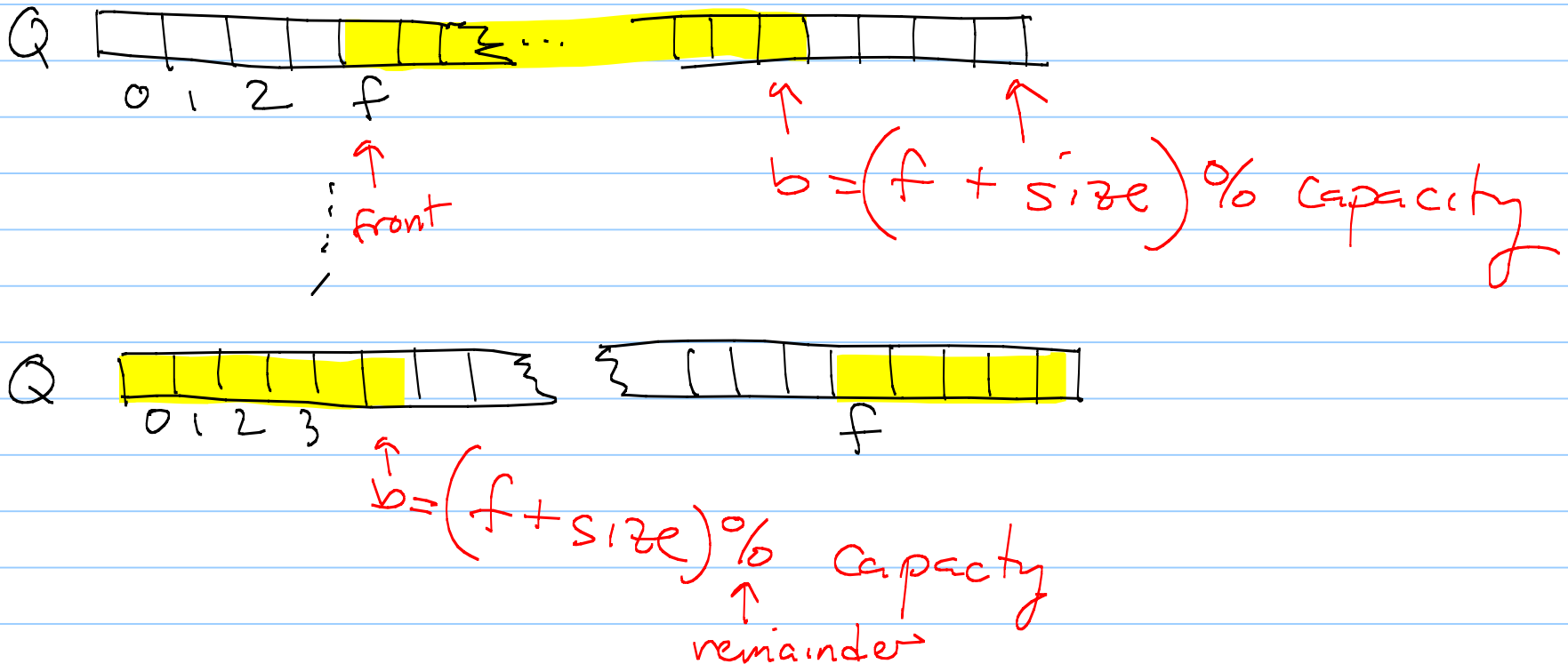
- linked version (doubly linked list, or circularly linked List)

- array

Private data:

```
Object * _Q;   <- (in constructor, will use new to make an array)
int _f;
int _size;
int _capacity;
```

Wrapping Around:



Two options:

- A lot of if statements

- Modular arithmetic: remainders

$4 \bmod 3 = 1$

$56 \bmod 6 = 2$

$3 \bmod 4 = 3$

Pseudocode

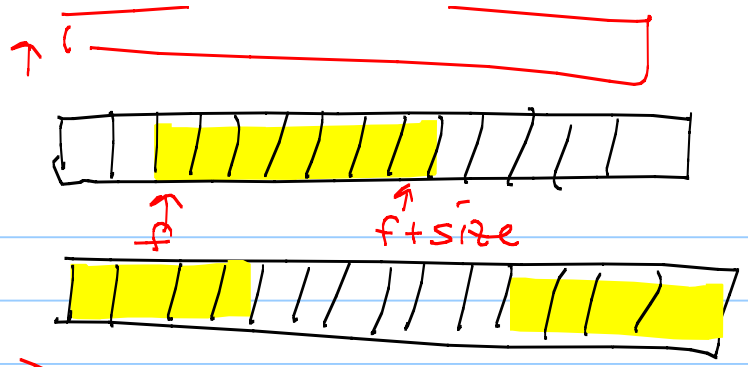
```
isEmpty():
    return (_size == 0)

size():
    return _size;
```

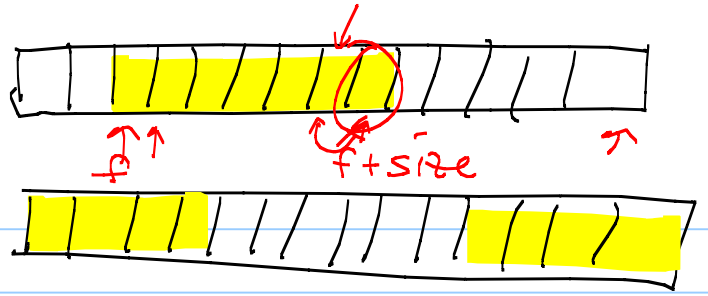
Constructor:

```
ArrayQueue(int cap = 1000) {
    _size = 0;
    _capacity = cap;
    _f = 0;
    _Q = new Object[_capacity];
}
```

```
top()
    return Q[f]
```




```
void push (const Object & element){
    if (_size == _capacity)
        throw runtime_error...
    _Q[(_f + _size) % _capacity] = element;
    _size ++;
}
```




```
void pop() {
    if (_size == 0)
        throw error

    _f = (_f+1) % _capacity;
    _size--;
}

Object top() {
    return _a[_f];
}
```

Housekeeping Functions

- Copy Constructor
- Destructor
- Operator =

(basically same as ArrayStack)

Actual code

(on webpage or in text)