Topic 16
Queues

"FISH queue: n.

[acronym, by analogy with FIFO (First In, First Out)] 'First In, Still Here'. A joking way of pointing out that processing of a particular sequence of events or requests has stopped dead. Also FISH mode and FISHnet; the latter may be applied to any network that is running really slowly or exhibiting extreme flakiness."

-The Jargon File 4.4.7

Queues

▸ Similar to Stacks

▸ Like a line

–In Britain people don't "get in line" they "queue up".

Queue Properties

‣ Queues are a first in first out data structure
  – FIFO (or LILO, but that sounds a bit silly)
‣ Add items to the end of the queue
‣ Access and remove from the front
  – Access to the element that has been in the structure the *longest* amount of time
‣ Used extensively in operating systems
  – Queues of processes, I/O requests, and much more

Queues in Operating Systems

▸ On a computer with 1 CPU, but many processes how many processes can actually use the CPU at a time?

▸ One job of OS, schedule the processes for the CPU

▸ issues: fairness, responsiveness, progress
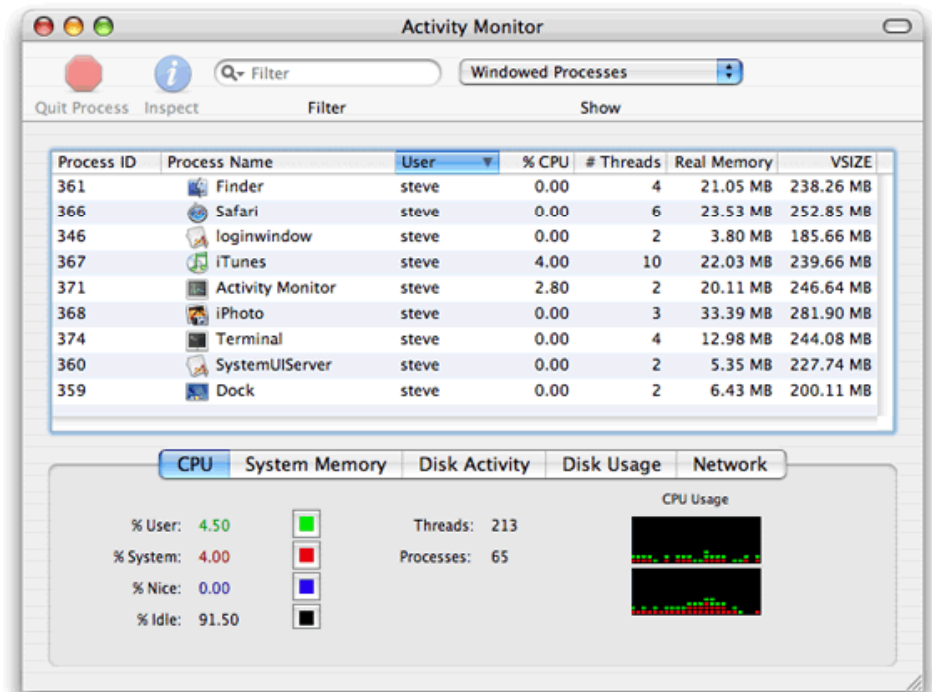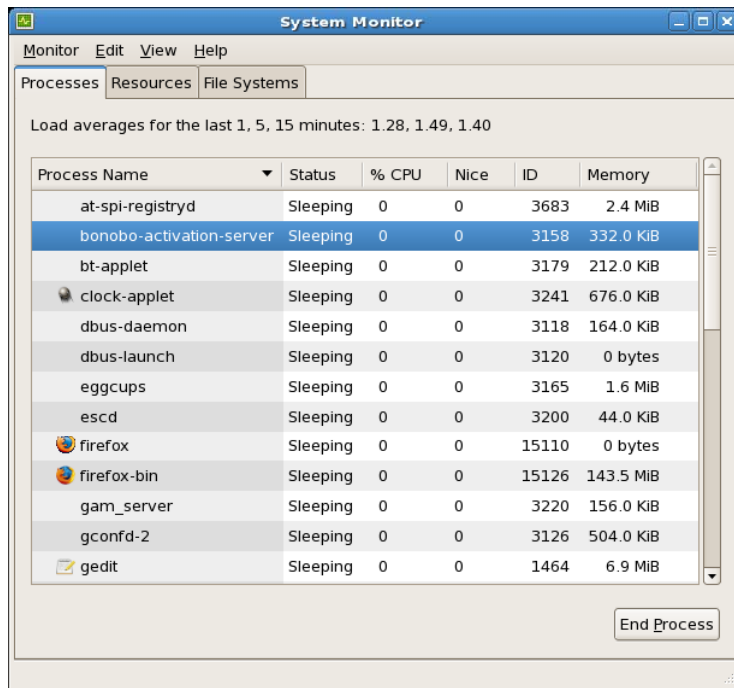
Queue operations

‣ `add(Object item)`

  – **a.k.a.** `enqueue(Object item)`

‣ `Object get()`

  – **a.k.a.** `Object front(), Object peek()`

‣ `Object remove()`

  – **a.k.a.** `Object dequeue()`

‣ `boolean isEmpty()`

‣ Specify in an interface, allow varied implementations

Queue interface, version 1

```
public interface Queue
{      //place item at back of this Queue
       enqueue(Object item);

       //access item at front of this queue
       //pre: !isEmpty()
       Object front();

       //remove item at front of this queue
       //pre: !isEmpty()
       Object dequeue();

       boolean isEmpty();
}
```

Implementing a Queue

‣ Given the internal storage container and choice for front and back of queue what are the Big O of the queue operations?

	ArrayList	LinkedList (Singly Linked)	LinkeList (Doubly Linked)
enqueue			
front			
dequeue			
isEmpty			

Attendance Question 1

‣ If implementing a queue with a singly linked list with references to the first and last nodes (head and tail) which end of the list should be the front of the queue in order to have all queue operations O(1)?
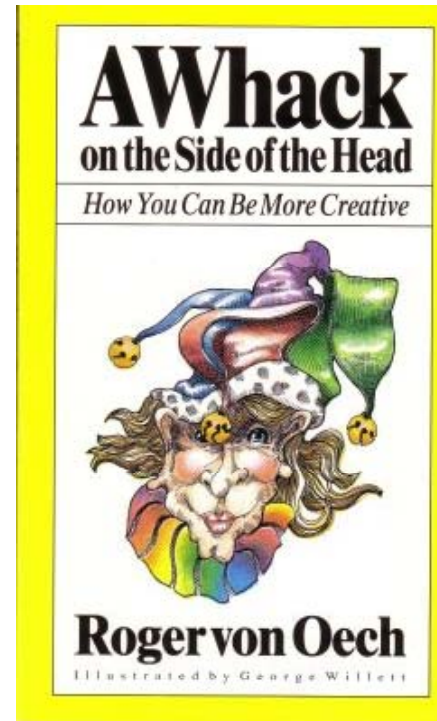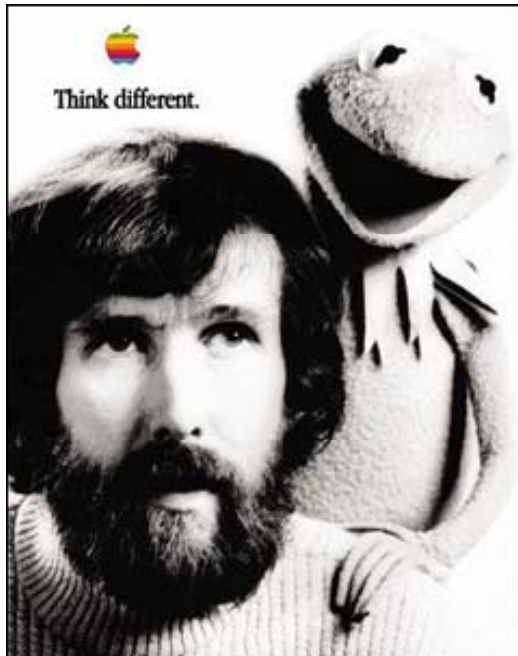
A. The front of the list should be the front of the queue

B. The back of the list should be the front of the queue.

C. D. E. I don't know, but I am sure looking forward to taking 307 again some time.

Alternate Implementation

‣ How about implementing a Queue with a native array?

– Seems like a step backwards

Application of Queues

‣ Radix Sort
  – radix is a synonym for *base.* base 10, base 2
‣ Multi pass sorting algorithm that ***only*** looks at individual digits during each pass
‣ Use queues as *buckets* to store elements
‣ Create an array of 10 queues
‣ Starting with the least significant digit place value in queue that matches digit
‣ empty queues back into array
‣ repeat, moving to next least significant digit

Radix Sort in Action: 1s

‣ original values in array

113, 70, 86, 12, 93, 37, 40, 252, 7, 79, 12

‣ Look at ones place

11<u>3</u>, 7<u>0</u>, 8<u>6</u>, 1<u>2</u>, 9<u>3</u>, 3<u>7</u>, 4<u>0</u>, 25<u>2</u>, <u>7</u>, 7<u>9</u>, 1<u>2</u>

‣ Queues:

0 7<u>0</u>, 4<u>0</u>   5

1   6 8<u>6</u>

2 1<u>2</u>, 25<u>2</u>, 1<u>2</u>   7 3<u>7</u>, <u>7</u>

3 11<u>3</u>, 9<u>3</u>   8

4   9 <u>9</u>, 7<u>9</u>

Radix Sort in Action: 10s

▸ Empty queues in order from 0 to 9 back into array

   70, 40, 12, 252, 12, 113, 93, 86, 37, 7, 9, 79

▸ Now look at 10's place

   7̲0, 4̲0, 1̲2, 25̲2, 1̲2, 11̲3, 9̲3, 8̲6, 3̲7, ̲7, ̲9, 7̲9

▸ Queues:

0	̲7, ̲9	5	25̲2
1	1̲2, 1̲2, 11̲3	6	
2		7	7̲0, 7̲9
3	3̲7	8	8̲6
4	4̲0	9	9̲3

Radix Sort in Action: 100s

‣ Empty queues in order from 0 to 9 back into array

7, 9, 12, 12, 113, 37, 40, 252, 70, 79, 86, 93

‣ Now look at 100's place

__7, __9, _12, _12, 113, _37, _40, 252, _70, _79, _86, _93

‣ Queues:

0 _7, _9, _12, _12, _40, _70, _79, _86, _93   5

1 113   6

2 252   7

3   8

4   9

Radix Sort in Action: Final Step

‣ Empty queues in order from 0 to 9 back into array

7, 9, 12, 12, 40, 70, 79, 86, 93, 113, 252

Radix Sort Code

```java
public static void sort(int[] list){
    ArrayList<Queue<Integer>> queues = new ArrayList<Queue<Integer>>();
    for(int i = 0; i < 10; i++)
        queues.add( new LinkedList<Integer>() );
    int passes = numDigits( list[0] );
    int temp;
    for(int i = 1; i < list.length; i++){
        temp = numDigits(list[i]);
        if( temp > passes )
            passes = temp;
    }
    for(int i = 0; i < passes; i++){
        for(int j = 0; j < list.length; j++){
            queues.get(valueOfDigit(list[j], i)).add(list[j]);
        }
        int pos = 0;
        for(Queue<Integer> q : queues){
            while( !q.isEmpty())
                list[pos++] = q.remove();
        }
    }
}
```